



TECHNISCHE
UNIVERSITÄT
DARMSTADT

LEARNING MODELS OF BEHAVIOR
FROM DEMONSTRATION AND THROUGH INTERACTION

Dem Fachbereich Elektrotechnik und Informationstechnik der
TECHNISCHEN UNIVERSITÄT DARMSTADT

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
vorgelegte Dissertation

von

ADRIAN ŠOŠIĆ, M.Sc.

Erstgutachter: Prof. Dr.-Ing. Abdelhak M. Zoubir
Zweitgutachter: Prof. Dr. techn. Heinz Koepl

Darmstadt 2018

Šošić, Adrian — Learning Models of Behavior From Demonstration and Through Interaction
Darmstadt, Technische Universität Darmstadt
Jahr der Veröffentlichung der Dissertation auf TUpriints: 2018
URN: urn:nbn:de:tuda-tuprints-81079
Tag der mündlichen Prüfung: 21.08.2018

Veröffentlicht unter CC BY-SA 4.0 International
<https://creativecommons.org/licenses/>

“ *Knowledge is probability.* ”

Anonymous

Acknowledgments

Many people have contributed to the success of this work in various ways and I would like to express my gratitude to all those who feel addressed.

I sincerely thank my supervisors Prof. Abdelhak M. Zoubir and Prof. Heinz Koeppel, who made this dissertation possible in the first place. Thank you for letting me be part of your research groups, for giving me the opportunity to work with so many outstanding people, and for supporting me during my doctoral studies. I would also like to thank Prof. Rolf Jakoby, Prof. Sebastian Schöps and Prof. Sascha Preu for being on my committee.

A huge thanks goes to all current and former members of the Signal Processing Group: Di Jin, Tim Schäck, Sergey Sukhanov, Stefano Fortunati, Michael Fauß, Afief Dias Pambudi, Patricia Binder, Freweyni Kidane Teklehaymanot, Nevine Demitri, Lala Khadidja Hamaidi, Renate Koschella, Hauke Fath, Fiky Suratman, Feng Yin, Sara Al-Sayed, Sahar Khawatmi, Mark Ryan Leonard, Wenjun Zeng, Stefan Leier, Mouhammad Alhumaidi, Michael Leigsnering, Wassim Suleiman, Michael Muma, Christian Debes, Gökhan Gül, Amare Kassaw and Roy Howard. A very special thanks goes to Simon Rosenkranz for being the most challenging and exhaustive conversationalist, to my former roommates Jürgen Hahn and Christian Weiss for teaching me how to tame a moving Dirac, to Dominik Reinhard for always keeping me hydrated, and to Ann-Kathrin Seifert for her superhuman eyesight and image processing skills.

In addition, I would like to thank the entire Bioinspired Communication Systems Lab: Dominik Linzner, Bastian Alt, Tim Prangemeier, Christian Wildner, Sikun Yang, Maleen Hanst, Nikita Kruk, Kilian Heck, Lukas Köhs, Leo Bronstein, Derya Altintan, Nurgazy Sulaimanov, Jascha Diemer, Francois-Xavier Lehr, Klaus-Dieter Voss, Markus Baier, Christine Cramer and, in particular, Wasiur Rahman Khuda Bukhsh for our joint efforts to win the Rupert Award.

Also, I would like to thank Maximilian Hüttenrauch, Prof. Gerhard Neumann, Prof. Elmar Rückert and Prof. Jan Peters for the pleasant cooperation.

Finally and most importantly, I would like to thank my parents Christel and Nino, my sister Anja, and my parents-in-law Andrea and FJ. Thank you for your support and always being there for me!

Thank you, Leo, for everything!

Darmstadt, 16.10.2018

Kurzfassung

Die vorliegende Dissertation befasst sich mit dem autonomen Erlernen von Verhaltensmodellen zur Beschreibung sequentieller Entscheidungsprozesse. Behandelt werden sowohl theoretische Aspekte der Verhaltensmodellierung — wie das Erlernen geeigneter Repräsentationen zur Abstraktion eines Entscheidungsprozesses — als auch praktische Schwierigkeiten bei der algorithmischen Umsetzung.

Die erste Hälfte der Dissertation beschäftigt sich mit dem Problem, eine gegebene Entscheidungsstrategie auf Basis von Beobachtungen zu generalisieren (engl.: *learning from demonstration*). Zur Lösung des Problems werden zwei unterschiedliche Modellierungsparadigmen vorgestellt: Zunächst wird ein nichtparametrischer Ansatz entwickelt, der es ermöglicht, zugrundeliegende Verhaltensmuster direkt auf Entscheidungsebene zu erfassen. Eine wesentliche Herausforderung in der Konzeption der Methodik besteht darin, auch im Fall einer unendlichen Anzahl von Systemzuständen möglichst wenige Annahmen über das beobachtete Systemverhalten zu treffen. Durch adaptive Anpassung der Modellordnung an die Komplexität der gezeigten Verhaltensmuster ist das vorgestellte Modell imstande, stochastische Entscheidungsstrategien jeglicher Art wiederzugeben. Als Nächstes wird ein nichtparametrischer Ansatz nach dem Prinzip des *inverse reinforcement learning (IRL)* konzipiert. Hierzu wird auf eine Form der Modellierung zurückgegriffen, bei welcher der Entscheidungsprozess in einzelne Teilprozesse untergliedert wird, um eine effiziente Rekonstruktion der Handlungsstrategie auf Intentionsebene zu ermöglichen. Im Gegensatz zu den meisten existierenden Methoden ist das Modell in der Lage, periodische Verhaltensmuster und Entscheidungsstrategien mit zeitabhängigen Zielen ohne zusätzliche Nachverarbeitungsschritte wiederzugeben. Aufgrund des modularen Aufbaus der Modelle bieten beide Paradigmen die Möglichkeit kompakte, an das Systemverhalten angepasste Darstellungen des Entscheidungsprozesses zu lernen. Die vorgestellten Modellierungsansätze werden anhand verschiedener Testszenarien evaluiert und mit existierenden Methoden verglichen. Hierzu werden sowohl synthetische Fallbeispiele als auch diverse Echtdatensätze herangezogen, die mit Hilfe eines KUKA Leichtbau-Roboterarms aufgenommen wurden.

In der zweiten Hälfte der Arbeit verlagert sich der Schwerpunkt auf Multiagentensysteme. Ziel ist die effiziente Modellierung der Entscheidungsprozesse in großen Agentennetzwerken mit homogener Architektur. Zunächst wird eine neue Klasse von Agentensystemen eingeführt, um die mathematische Grundlage zur Beschreibung verteilter homogener Systeme zu schaffen. Für diese Systemklasse wird das IRL Problem diskutiert und ein

Meta-Lernalgorithmus entwickelt, der zur Lösung explizit die Symmetrien des Systems nutzt. Im Zuge dessen wird ein heterogenes Lernschema vorgestellt, welches das kollektive Systemverhalten auf Basis lokaler Zustandsbeobachtungen optimiert. Im letzten Teil der Arbeit wird schließlich eine Kontinuum-Beschreibung des Modells hergeleitet, welche die Simulation des Netzwerks für große Agentenzahlen ermöglicht. Zu diesem Zweck werden die entsprechenden Kontinuum-Systemkomponenten und Optimalitätskriterien eingeführt. Um das Prinzip der Kontinuum-Modellierung zu veranschaulichen, werden abschließend mehrere Beispiele kollektiver Entscheidungsfindung aufgeführt, welche die Vorteile gegenüber einer agentenbasierten Verhaltensmodellierung aufzeigen.

Abstract

This dissertation is concerned with the autonomous learning of behavioral models for sequential decision-making. It addresses both the theoretical aspects of behavioral modeling—like the learning of appropriate task representations—and the practical difficulties regarding algorithmic implementation.

The first half of the dissertation deals with the problem of *learning from demonstration*, which consists in generalizing the behavior of an expert demonstrator based on observation data. Two alternative modeling paradigms are discussed. First, a nonparametric inference framework is developed to capture the behavior of the expert at the policy level. A key challenge in the design of the framework is the objective of making minimal assumptions about the observed behavior type while dealing with a potentially infinite number of system states. Due to the automatic adaptation of the model order to the complexity of the shown behavior, the proposed approach is able to pick up stochastic expert policies of arbitrary structure. Second, a nonparametric inverse reinforcement learning framework based on subgoal modeling is proposed, which allows to efficiently reconstruct the expert behavior at the intentional level. Other than most existing approaches, the proposed methodology naturally handles periodic tasks and situations where the intentions of the expert change over time. By adaptively decomposing the decision-making problem into a series of task-related subproblems, both inference frameworks are suitable for learning compact encodings of the expert behavior. For performance evaluation, the models are compared with existing frameworks on synthetic benchmark scenarios and real-world data recorded on a KUKA lightweight robotic arm.

In the second half of the work, the focus shifts to multi-agent modeling, with the aim of analyzing the decision-making process in large-scale homogeneous agent networks. To fill the gap of decentralized system models with explicit agent homogeneity, a new class of agent systems is introduced. For this system class, the problem of inverse reinforcement learning is discussed and a meta learning algorithm is devised that makes explicit use of the system symmetries. As part of the algorithm, a heterogeneous reinforcement learning scheme is proposed for optimizing the collective behavior of the system based on the local state observations made at the agent level. Finally, to scale the simulation of the network to large agent numbers, a continuum version of the model is derived. After discussing the system components and associated optimality criteria, numerical examples of collective tasks are given that demonstrate the capabilities of the continuum approach and show its advantages over large-scale agent-based modeling.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Aims, Contributions and Thesis Overview	4
1.3	Related Work	7
1.4	Publications	8
2	Fundamentals	11
2.1	Sequential Decision-Making	11
2.2	Learning from Demonstration	22
I	Single-Agent Systems: Subintentional Modeling	29
3	Policy Recognition: A Simple Approach	29
3.1	Maximum Likelihood Estimation	31
3.2	Maximum A Posteriori Estimation	32
3.3	Experimental Results	34
3.4	The Next Steps	37
4	Parametric Policy Recognition	38
4.1	Finite State Spaces: The Static Model	39
4.2	Toward Large State Spaces: A Clustering Approach	44
4.3	Countably Infinite and Uncountable State Spaces	50
5	Nonparametric Policy Recognition	54
5.1	A Dirichlet Process Mixture Model	55
5.2	Policy Recognition using the ddCRP	57
6	Experimental Results	60
6.1	Uncountable State Space	60
6.2	Finite State Space	66
7	Summary	69
II	Single-Agent Systems: Intentional Modeling	71
8	The Subgoal Principle	72
9	Bayesian Nonparametric Inverse Reinforcement Learning	76
9.1	Revisiting the BNIRL Framework	76

9.2	Limitations of BNIRL	79
10	Nonparametric Spatio-Temporal Subgoal Modeling	84
10.1	The Subgoal Likelihood Model	85
10.2	Modeling Time-Invariant Intentions	92
10.3	Modeling Time-Varying Intentions	94
10.4	Prediction and Inference	96
11	Experimental Results	102
11.1	Proof of Concept	104
11.2	Random MDP Scenario	105
11.3	Robot Experiment	108
11.4	Active Learning	113
12	Summary	115
III	Multi-Agent Systems: Swarm Modeling	117
13	Motivation	117
14	The swarMDP Model	119
15	Inverse Reinforcement Learning in Swarm Systems	121
15.1	Policy Update	122
15.2	Value Estimation	128
15.3	Reward Update	129
16	Experimental Results	130
16.1	The Vicsek Model	130
16.2	The Ising Model	133
17	Summary	136
IV	Multi-Agent Systems: Continuum Modeling	139
18	Motivation	140
19	The Agent Model	142
19.1	System Dynamics	142
19.2	Observation Model	143
19.3	System Controller	145
19.4	Reward Models	147
19.5	Value Estimation	149
19.6	The Agent Model in the swarMDP Context	150

20 The Continuum Model	151
20.1 The Continuum Equation	151
20.2 Reward Models	152
20.3 Value Estimation	154
20.4 Continuum Modeling versus Agent-Based Modeling	154
21 Reinforcement Learning in the Continuum Model	156
22 Experimental Results	159
22.1 Aggregation	161
22.2 Partial Observability	161
22.3 Aggregation and Localization	166
23 Summary	169
 Discussion and Outlook	 171
 Appendices	 177
A Marginal Invariance and Policy Prediction	179
B KUKA Robot Experiment	181
C Derivation of the Continuum Equation	185
 List of Acronyms	 189
 References	 191

Introduction

The dissertation at hand addresses one of the key challenges in intelligent systems design: the behavioral modeling of other actors. The work presents a coherent framework for the autonomous construction of behavioral representations based on experience data, using the Markov decision process formalism as a formal language for sequential decision-making.

The focus of the work lies on two different modeling scenarios. The first is *single-agent behavioral modeling*, which is concerned with analyzing and predicting the decisions of an observed demonstrator. The scenario occurs in a wide variety of application areas, such as human-machine interaction, behavioral psychology, or financial modeling, where behavioral representations are used, e.g., for plan recognition, preference elicitation, credit scoring and risk modeling, or fraud detection. The second scenario is concerned with the *behavioral modeling of large-scale networks* that are formed by homogeneous agent populations, such as animal flocks, robot swarms or sensor networks. In this scenario, the focus is not on the accurate modeling of the individual agent but lies on the reconstruction of the network mechanics, to provide insights into the collective behavior that emerges through the interactions of agents.

An important aspect of the work is the systematic consideration of uncertainty about the observed system processes by means of statistical inference. Hence, the presented methodology is built upon probabilistic system models that allow to account for the stochasticity in the system environment and the randomness of the agents' decisions. The developed approaches are demonstrated in numerous simulated and real-world experiments, which are representative for a variety of application scenarios. An overview of possible use cases is provided in the next section, followed by a summary of contributions and an outline of the work.

1.1 Motivation

Behavioral modeling is an integral part of many scientific disciplines such as economics, sociology, psychology, biology, robotics, electrical engineering and computer science. In the financial sector, for example, high frequency trading has become an established business field, where computers analyze market situations based on the behavior of

		<u>System Type</u>	
		Single-Agent System	Multi-Agent System
Objective	Analysis	Behavioral Psychology	Swarm Biology
	Imitation	Robot Learning	Routing Networks
	Interaction	Human-Machine Interaction	Financial Modeling

Table 1.1: Some applications of behavioral modeling.

other traders and trigger transactions in the range of a millisecond to gain an advantage over competitors [Yan+12; BT03]. Similarly, credit card companies analyze the purchasing behavior of consumers in order to estimate their default risk [THS01; ST11]. In autonomous driving, models of pedestrian behavior are used to facilitate a realistic assessment of the scene and avoid hazardous situations at an early stage [PYG09; Gu+16].

Among the many possible reasons for the use of behavioral models, three main motivations stand out that are common to most application scenarios (see Table 1.1):

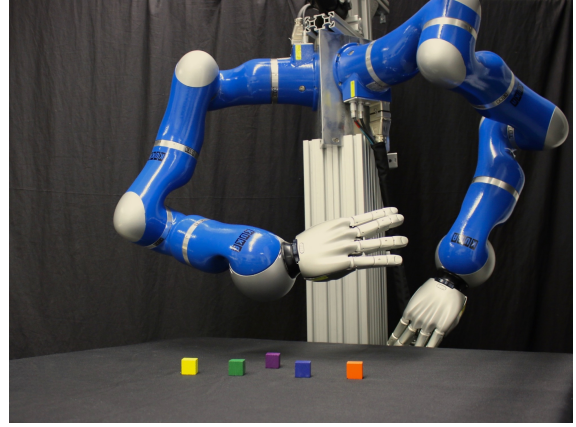
- (i) the *analysis* of a given system behavior,
- (ii) the *imitation* of an observed strategy, and
- (iii) the *interaction* with the target system.

A brief overview of potential use cases and existing application areas is given below.

Behavior Analysis Behavioral models are found in all scientific fields that are concerned with behavior *analysis*. Classical examples are sociology and behavioral psychology, where behavioral representations are used for the discovery of patterns in social interactions and the identification of individual preferences [Gra78; RD11]. Similarly, behavioral models have been developed in biology and statistical physics to investigate the collective dynamics and emergent properties of large-scale systems [VZ12; CFL09]. These models are used, for example, to study collective animal behavior [Buh+06] (Figure 1.1a) or to analyze crowd phenomena and evacuation dynamics for disaster prevention [HJ09]. However, behavior analysis is not only important in social and natural sciences but is also required in many branches of technology. The range of applications is vast and includes examples such as traffic monitoring [MT13], cognitive radio/multiple access communication [Hay05], and dynamic scene understanding [Bux03].



(a) an ant colony



(b) a robotic arm

Figure 1.1: Two examples that illustrate the challenges and wide applicability of behavioral modeling in nature and technology. (a) An example of a homogeneous large-scale agent network. (b) An example of a single-agent system with many degrees of freedom.

Behavior Imitation In an engineering context, behavioral models are often used for *imitating* a certain decision-making strategy, with the objective to evoke similar behavioral characteristics as observed in the original system. For instance, modern routing protocols emulate the swarming behavior of self-organizing biological systems, such as ant colonies (Figure 1.1a) and fish schools, to solve collective tasks like shortest path finding or to improve the network’s energy efficiency and resilience against changes in the environment [ZAS12; Cam+06]. Similar concepts of skill transfer are found in other disciplines, like in robotics (Figure 1.1b), where human behavior is copied for solving complex control problems and learning new motor primitives [Arg+09; Osa+18].

System Interaction While behavioral models offer great possibilities for system identification and imitation, they can be also used as an instrument for learning how to *interact* with a given system. This opens up numerous possibilities for human-machine interaction. A concrete example is human-robot interaction, where behavioral models can serve as an interface for cooperation. For instance, by observing a user’s behavior over an extended period of time, a service robot can learn to anticipate the user’s intention and assist him or her in a given situation [FND03; Amo+14]. Another field of application are recommender systems, which model a user’s behavior and preferences in order to make suitable action suggestions for future decisions [AT05]. In a medical context, this can be used, for instance, to encourage the physical activity of a patient through personalized feedback [Hoc+16]. Further examples, which offer numerous applications, are content recommendation and information retrieval [LPE00; SK09].

1.2 Aims, Contributions and Thesis Overview

Generative probabilistic frameworks like Markov decision processes provide powerful tools for behavioral modeling as they allow to analyze the strategy and intentions of an agent at each stage of a decision-making process. At the same time, they offer the possibility to consider various sources of uncertainty about the agent behavior and its environment. From a practical perspective, however, the downside is that there is often not enough observation data available to accurately determine all model parameters, which makes a generalization of the observed behavioral strategy difficult. Moreover, computing exact solutions to most inference problems in these models is computationally infeasible. Particularly challenging are problems that require consideration of the agent’s planning process, which usually involves finding solutions to the underlying decision-making problem. Even if these solutions are tractable for a particular problem at hand, most inference approaches break down as soon as other decision-makers enter the field.

The goal of this thesis is to develop solution concepts that address these difficulties in a principled way, i.e., through the use of approximate inference techniques, structural assumptions about the agent’s task representation, and large-scale asymptotics. The presented methodology is based on two complementary learning principles (Chapter 2):

- (i) **learning from demonstration (LfD)**, which allows to construct behavioral representations of a task based on observation data, and
- (ii) **reinforcement learning (RL)**, which provides the possibility to optimize a behavioral strategy based on feedback from the underlying system process.

In summary, the thesis covers four main contributions:

- **Nonparametric Policy Recognition:** We develop a probabilistic inference framework for learning subintentional behavioral models from demonstrated trajectory data. The framework relies on minimal prior assumptions about the demonstrator’s behavior, which makes it possible to handle arbitrary types of stochastic expert policies. The nonparametric character of the model gives the flexibility to automatically adjust the complexity of the learned behavior representation to the diversity of the observed behavioral patterns, obviating the need to specify the complexity class of considered expert policies a priori. By addressing the inference problem on the subintentional level, the framework bypasses the necessity to invert the planning process of the demonstrator, which enables a straightforward extension to infinite domains.

- **Nonparametric Subgoal Modeling:** Building upon on the proposed subintentional approach, we develop a nonparametric inverse reinforcement learning framework that allows to reason about the latent subgoals of the demonstrator, leveraging the inference to the intentional level. The framework targets a data-efficient use of the expert demonstrations by taking into account the context information at the observed decision times. The context information is processed with the help of a state space metric based on stopping times, which is derived naturally from the applied subgoal principle. To improve the subgoal localization accuracy, the context-aware model formulation is combined with a redesigned likelihood model that reflects the action selection process of an imperfect goal-oriented agent more accurately. Finally, a variant of the model is presented that allows to handle situations where the intentions of the demonstrator change over time.
- **Inverse Reinforcement Learning in Swarm Systems:** A generalized inverse reinforcement learning procedure is designed for modeling the local reward mechanism in homogeneous agent networks. To this end, a new model class of agent networks is introduced that reflects the characteristics of natural swarm systems. For this model class, the necessary optimality conditions are defined and a meta learning algorithm is presented that can be instantiated using existing inverse reinforcement learning algorithms. To solve the associated policy estimation problem, a heterogeneous learning scheme is presented that facilitates the state exploration of the agents by artificially breaking the system symmetries.
- **Large-Scale Decision-Making via Continuum Modeling:** In order to address the computational challenges of simulating large-scale agent networks, a continuum formulation of the swarm model is derived. To this end, the system dynamics are reformulated in the continuous-time domain and the corresponding network mechanism are discussed. For the purpose of reinforcement learning, the effect of centralized and decentralized reward feedback is investigated. The tight relationship with the introduced swarm model class offers the possibility to use the framework as a computational tool for large-scale behavioral inference.

The four main contributions are reflected in the structure of the thesis, which is organized accordingly into four parts. A high-level view is provided in Figure 1.2, which shows the connections between the discussed topics and serves as a roadmap for further reading. A short outline of the content is as follows:

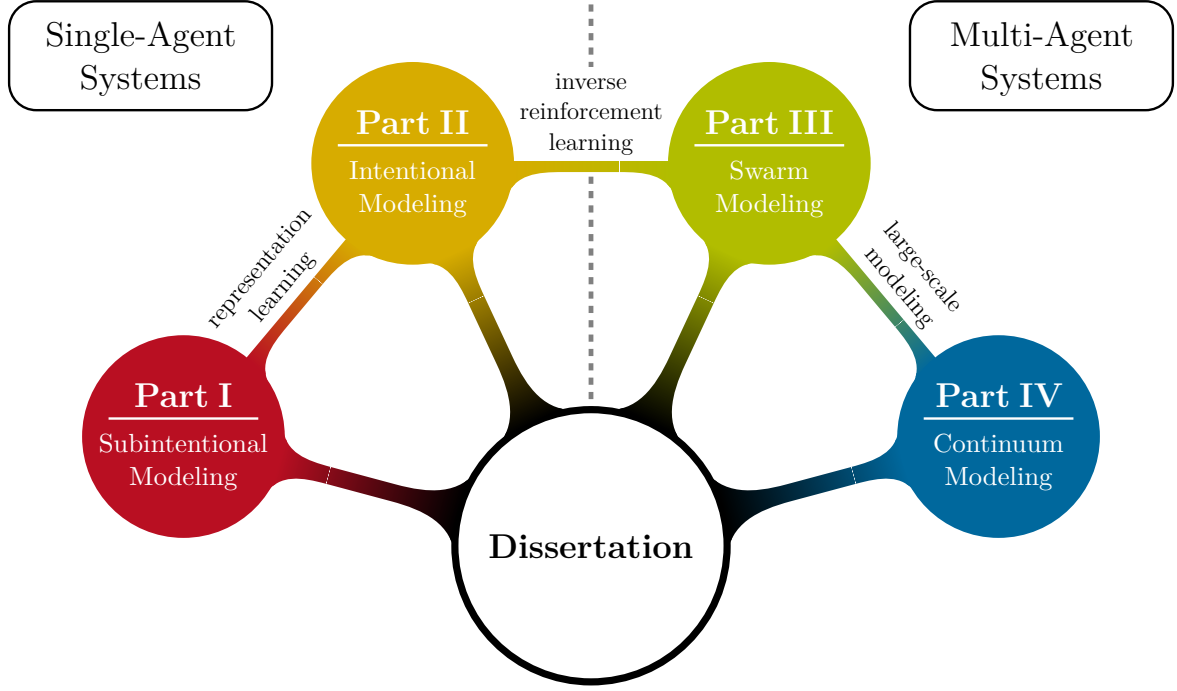


Figure 1.2: Graphical overview of the covered topics and their relationships.

Chapter 2 gives a brief introduction to the theory of sequential decision-making and reviews the basic modeling concepts in this field. Furthermore, the chapter provides an overview on the subject of learning from demonstration.

Part I starts with a subintentional view on the LfD problem. Based on the simplest possible LfD scenario—an agent executing a deterministic time-invariant policy in a finite domain—we discuss the key challenges and the role of prior information in the learning problem. By gradually adding more levels of difficulty to the scenario, we develop, step by step, a Bayesian inference framework that solves the LfD objective through an appropriate task-adapted partitioning of the system state space. The working principle is illustrated with an example scenario before the framework is tested against existing methods on a set of benchmark problems. The presented concepts are taken from [ŠZK16; ŠZK18a].

In **Part II**, we transfer the partitioning concept from Part I to the intentional level, which yields a subgoal-based inference framework for nonparametric inverse reinforcement learning. To arrive at the final model formulation, the weaknesses of existing subgoal frameworks are identified and subsequently eliminated through an appropriate model redesign. To demonstrate the advantages over existing approaches, the resulting framework is eval-

uated on various benchmark data sets and on real-world demonstration data recorded on a robotic arm. The content of this part is based on the findings in [ŠZK18b; Šoš+18].

In **Part III**, we move over to the multi-agent domain and tackle the inverse reinforcement learning problem in homogeneous agent networks. For this purpose, we design an inference method that exploits the inherent homogeneity property of the system and allows to efficiently handle large network sizes. Experimental results are presented for two different system types that cover both static and dynamic agent neighborhoods. The underlying ideas were originally published in [Šoš+17a; Šoš+17b].

In **Part IV**, we focus on settings where the network size is too large to be handled in form of agent-based simulations. To approximate the system dynamics in these regimes, we introduce an alternative model formulation, based on a continuum description of the system’s state. The method is tested in a series of experiments, which demonstrate the potentials of continuum-based modeling and show its advantages over large-scale agent simulation. The presented ideas stem from [ŠZK18c].

Finally, we conclude the thesis with a discussion of the presented ideas and an outlook for future work.

1.3 Related Work

The contributions in this thesis cover topics from various disciplines. In order to provide a structured overview of the related literature, relevant works are discussed in their respective context:

- For a general overview of research in the area of learning from demonstration, see Section 2.2.
- For a summary of existing approaches related to subtask modeling, see Chapter 8.
- For related work in the field of multi-agent inverse reinforcement learning, see Chapter 13.
- For a literature review on large-scale modeling of agent systems, see Chapter 18.

1.4 Publications

The following publications have been produced during the period of doctoral candidacy:

Internationally Refereed Journal Articles

- [Šoš+18] A. Šošić, E. Rückert, J. Peters, A. M. Zoubir, and H. Koepl. “Inverse reinforcement learning via spatio-temporal nonparametric subgoal modeling”. In: *Journal of Machine Learning Research* (2018). **Accepted**. arXiv: 1803.00444
- [ŠZK18c] A. Šošić, A. M. Zoubir, and H. Koepl. “Reinforcement learning in a continuum of agents”. In: *Swarm Intelligence* 12.1 (2018), pp. 23–51
- [ŠZK18a] A. Šošić, A. M. Zoubir, and H. Koepl. “A Bayesian approach to policy recognition and state representation learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.6 (2018), pp. 1295–1308

Internationally Refereed Conference and Workshop Papers

- [ŠZK18b] A. Šošić, A. M. Zoubir, and H. Koepl. “Inverse reinforcement learning via nonparametric subgoal modeling”. In: *AAAI Spring Symposium on Data-Efficient Reinforcement Learning*. 2018
- [HŠN18] M. Hüttenrauch, A. Šošić, and G. Neumann. “Local communication protocols for learning complex swarm behaviors with deep reinforcement learning”. In: *International Conference on Swarm Intelligence*. **Accepted**. 2018
- [Šoš+17b] A. Šošić, W. R. KhudaBukhsh, A. M. Zoubir, and H. Koepl. “Inverse reinforcement learning in swarm systems”. In: *International Conference on Autonomous Agents and Multiagent Systems*. **Best paper award finalist**. 2017, pp. 1413–1421
- [Šoš+17a] A. Šošić, W. R. KhudaBukhsh, A. M. Zoubir, and H. Koepl. “Inverse reinforcement learning in swarm systems”. In: *AAMAS Workshop on Transfer in Reinforcement Learning*. 2017
- [HŠN17] M. Hüttenrauch, A. Šošić, and G. Neumann. “Guided deep reinforcement learning for swarm systems”. In: *AAMAS Workshop on Autonomous Robots and Multirobot Systems*. 2017

- [HŠN16] M. Hüttenrauch, A. Šošić, and G. Neumann. “Guided deep reinforcement learning for swarm systems”. In: *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems*. 2016
- [ŠZK16] A. Šošić, A. M. Zoubir, and H. Koepl. “Policy recognition via expectation maximization”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2016, pp. 4801–4805
- [Gut+14] T. Guthier, A. Šošić, V. Willert, and J. Eggert. “sNN-LDS: spatio-temporal non-negative sparse coding for human action recognition”. In: *International Conference on Artificial Neural Networks*. 2014, pp. 185–192

2

Fundamentals

The goal of this chapter is to provide a concise introduction to the theory of Markov decision processes, which form the mathematical foundation for the ideas presented in this thesis. In the first part of the chapter, we begin with a short refresher on sequential decision-making, which leads us, step by step, to the two basic solution strategies in this field: probabilistic planning and reinforcement learning. As there are many excellent textbooks on the subject of sequential decision-making, we focus on the key aspects of the problem, without delving into all of its numerous facets. Details that are specific to the scenarios addressed in this thesis will be provided later in the main parts.

In the second half of the chapter, we briefly touch on the principle of learning from demonstration, which represents one of the central concepts in the thesis. After a short overview of the topic, we discuss the two fundamental views on the learning problem, i.e., intentional and subintentional behavioral modeling, and finish with a discussion of the advantages and disadvantages of both approaches.

Although this chapter serves as a self-contained introduction to the presented work, the reader should be aware that it does *not* cover all basics that are necessary to understand the contributions of this thesis in every detail, and that later sections build upon ideas from other disciplines that are not discussed here. For example, throughout the thesis, we will make extensive use of graphical models [KF09] as a modeling language to describe the statistical relationships of our model variables. Also, the reader will encounter some of the cornerstones from the theory of Bayesian nonparametrics, like the Chinese restaurant process and friends [Hjo+10; BF11]. However, in the context of this work, it is sufficient to consider these concepts as modeling tools, as we will not leave the scope of the theory that can be found in the respective standard literature.

2.1 Sequential Decision-Making

Sequential decision-making (SDM) is the process of solving a task that requires the execution of *a series of consecutive actions*. The number of possible examples is huge and includes all problems that involve a sequence of temporally dependent events—ranging from simple daily routines like washing laundry or preparing a meal, up to complex tasks like the control of a factory plant or the conclusion of a political agreement. The

action takers involved in these tasks—commonly referred to as *agents*—can be all kinds of entities that are capable of making decisions. This certainly includes human or animal beings, but also abstract entities with decision-making capabilities like political bodies or computer programs.

The crux, which makes sequential decision-making a hard problem, is that each action taken by the agent(s) affects, sometimes irrevocably, the circumstances at the subsequent decision times. For instance, in a chess game, a single move can determine whether a player wins or loses a game. Because of this property, finding optimal solution strategies to SDM problems requires a proper long-term planning of events, which often requires temporary acceptance of unfavorable situations in order to achieve greater overall success.

In the following sections, we provide a short overview of the most important aspects of sequential decision-making. We begin with a formal definition of Markov decision processes, which provide the necessary mathematical formalism to model all kinds of sequential problems.

2.1.1 Markov Decision Processes

The Markov decision process (MDP) forms the heart of all SDM models: it is the fundamental building block that captures the sequential nature of a decision-making process. In the course of this thesis, we will encounter different extensions of the basic MDP formalism, which all build upon the same key elements discussed in this section.

In the literature, one can find various closely related definitions of MDPs that differ slightly in their construction but essentially all capture the same spirit. Following the style in [KLM96; LDK95], we define an MDP as a quadruple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R)$, where \mathcal{S} and \mathcal{A} denote, respectively, the *state space* and *action space* of the process, \mathcal{T} defines its *transition dynamics*, and R is a *reward model*. The meaning of these elements is the following:

State Space The state space \mathcal{S} is the collection of all possible situations that can occur to an agent during the decision-making process. These situations, called *states*, are determined by both the external circumstances of the environment as well the internal conditions of the agent, such as the agent’s current belief about the world. By definition, a state provides the complete summary information needed to reason about all contingencies that can occur by taking future actions (provided that the dynamics of the environment are known). In mathematical terms, this characteristic is expressed by the *Markov property*, which states that the future evolution of a system is

conditionally independent of its past given the present. For simplicity, we will assume that \mathcal{S} is finite in this chapter; more complex scenarios that involve infinite state spaces will be discussed in the main parts of the thesis.

Action Space The action space \mathcal{A} represents the set of possible decisions or *actions* through which the agent can interact with its environment. Each action can be regarded as a particular response from the agent to its current situation that is carried out to pursue a certain goal. The mathematical details of this action selection mechanism are explained in Section 2.1.2. For most parts of the thesis, we will assume that the action space of the agent is finite, but we note that many of the ideas presented apply to the infinite case as well.

Transition Dynamics The transition dynamics \mathcal{T} define the *effect* of a decision on the agent’s state, i.e., they describe what happens to the world when the agent takes a particular action. In real-world scenarios, for example, \mathcal{T} could describe the physical circumstances that underlie a given system environment, while, in a game, \mathcal{T} could encode the rules that define the consequences of a player’s move. In the thesis, we generally consider *stochastic* transition models $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$, i.e., we assume that the effect of an action on the environment is random, and we write $\mathcal{T}(s' | s, a)$ to denote the probability (density) associated with the event that the agent reaches state $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. While we typically do not distinguish between a real system and its mathematical description in form of an MDP, it is important to keep in mind that \mathcal{T} is only a *model* of the transition dynamics of an environment and that the laws of the actual system may be different (see, for example, Section 6.2.3).

Reward Model The reward model R plays the central role in any MDP because it defines (implicitly) *the task* for the agent. Inspired by the biological process of learning from reward/punishment [Doy07] (c.f. *operant conditioning* [DB02]), the reward model provides the agent with an instantaneous feedback about its current situation in the context of the executed task. The received information, given in form of a real-valued discrete-time signal, can then be used by the agent to adapt its behavior and react appropriately to the circumstances imposed by the environment. In this work, we restrict ourselves to models that provide the reward as a *deterministic function* of the agent’s state, i.e., $R : \mathcal{S} \rightarrow \mathbb{R}$. Other definitions found in the literature include reward models that additionally depend on the executed action/the successor state of the agent. However, these variants are mathematically equivalent to the above-mentioned definition as the same model class can be described through an appropriate extension of the agent’s state.

2.1.2 The Policy

The MDP model introduced in Section 2.1.1 provides all ingredients that are necessary to *formulate* a decision-making problem: the state space \mathcal{S} contains all possible situations that can occur to an agent, the action space \mathcal{A} defines the degrees of freedom the agent has, the transition dynamics \mathcal{T} define the system environment, and the reward model R sets the task. However, the resulting model does not prescribe *how* the given task is to be solved by the agent. Missing is an action rule—called a *policy*—that tells the agent how to behave in a given situation.

Depending on the problem at hand, that action rule can take on different forms of varying complexity. In the most general setting, it can be necessary for an agent to consider the entire history of previous events in order to make informed decisions that optimally solve a problem, leading to increasingly complex policy models with every further decision. Yet, in Section 2.1.1 we argued that the current state of the agent comprises all information that is relevant to plan future steps. Hence, assuming that the Markov property holds and that the agent is perfectly aware of its situation, it is sufficient to consider the reduced class *reactive policies* (also referred to as *Markov policies*), which determine the next action based on the current state only.* In Section 2.1.5, we point to a fundamental result from decision-making theory, which gives the formal mathematical justification for this important statement. In the second half of the thesis, we also consider problems where the requirement of complete state information is violated.

In the following, we consider both *deterministic* Markov policies, which define a fixed mapping from states to actions, i.e., $\pi : \mathcal{S} \rightarrow \mathcal{A}$, as well as *stochastic* Markov policies, given as conditional distributions over actions, i.e., $\pi : \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. To keep our notation simple, we use the same symbol π for both policy types; which type is meant will be always clear from the context. Whenever we need to make the arguments of the policy explicit, we write $a = \pi(s)$ in the deterministic case and $a \sim \pi(a | s)$ in the stochastic case, where $s \in \mathcal{S}, a \in \mathcal{A}$.

Combining a policy with an MDP completes the decision-making model in the sense that it allows us to execute/simulate a certain behavior in a given system environment. More specifically, by telling the agent which action to execute at what state, the policy induces a random sequence $(s_1, a_1, r_1, s_2, a_2, r_2, \dots)$ of state-action-reward triplets according to the following procedure: starting at some initial state $s_1 \in \mathcal{S}$, the agent chooses an

* Note that the assumption is trivially fulfilled if we define the state variable as the agent's history.

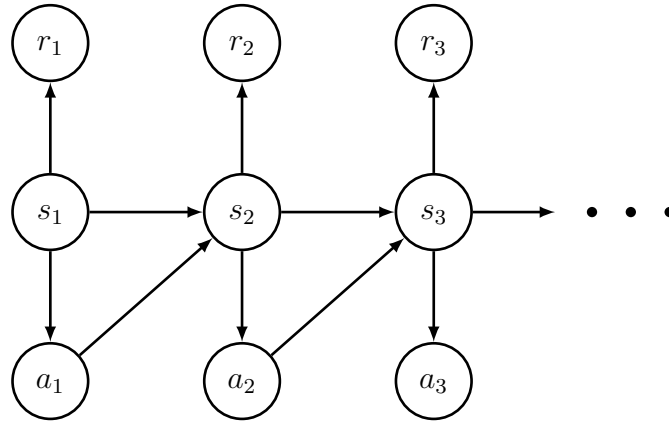


Figure 2.1: Graphical representation of a Markov decision process.

action $a_1 \in \mathcal{A}$ and transitions to a new state $s_2 \sim \mathcal{T}(s_2 | s_1, a_1)$ according to the system dynamics \mathcal{T} , where the next action $a_2 \in \mathcal{A}$ is selected that triggers another transition $s_3 \sim \mathcal{T}(s_3 | s_2, a_2)$, and so on. At each decision time t , the corresponding action a_t is determined or generated randomly according to the agent's policy, i.e., $a_t = \pi(s_t)$ or $a_t \sim \pi(a_t | s_t)$, respectively. Additionally, the agent receives feedback for every encountered state in form of an instantaneous reward, i.e., $r_t = R(s_t)$. Mathematically, this procedure defines a stochastic process on the product space $\mathcal{S} \times \mathcal{A} \times \mathbb{R}$, indexed by the natural numbers representing the decision times. The dependency structure of the involved random variables is visualized by the graphical model shown in Figure 2.1.

2.1.3 Value Functions and Bellman Recursion

Just as there may exist different paths to one and the same destination, there are usually several ways to accomplish an SDM task.* Having formally defined the action selection process of an agent, the next step is hence to quantify *how efficient* a policy is in a given context. For this purpose, we need a performance criterion that allows to assess the behavior of the agent.

Based on the stochastic process described in Section 2.1.2, a natural way to measure performance is by cumulating the (expected) future reward that is received by the agent. One such performance measure is given by the *infinite-horizon discounted optimality*

* The problem of optimal path finding can be in fact interpreted as a specific kind of SDM problem, known as the *(stochastic) shortest path problem* [BT91].

criterion [SB98], which is defined as

$$\begin{aligned} V^\pi(s) &\triangleq \mathbb{E} \left[r_1 + \gamma r_2 + \gamma^2 r_3 + \dots \mid s_1 = s, \pi \right] \\ &= \mathbb{E} \left[R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \dots \mid s_1 = s, \pi \right]. \end{aligned} \quad (2.1)$$

Herein, $\gamma \in [0, 1)$ represents a *discount factor*, which is used to weigh the importance of immediate against long-term reward. Note that there exist other popular choices, like the *infinite-horizon average criterion* and the *finite-horizon criterion* (see Part IV).

As common in the reinforcement literature, we refer to V^π as the *value function* (or *state value function*) of a policy π , while we call the inner part of the expectation in Equation (2.1) the *return* of the policy. Using the law of iterated expectations, it is possible to establish a recursive relationship for the value function, where the value at any state s is represented in terms of the (expected) value at the successor state s' , i.e.

$$\begin{aligned} V^\pi(s) &= \mathbb{E} \left[R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \dots \mid s_1 = s, \pi \right] \\ &= R(s) + \gamma \cdot \mathbb{E} \left[R(s_2) + \gamma R(s_3) + \gamma^2 R(s_4) + \dots \mid s_1 = s, \pi \right] \\ &= R(s) + \gamma \cdot \mathbb{E}_{s' \sim \mathcal{T}(s' \mid s, \pi(s))} \left[V^\pi(s') \right]. \end{aligned} \quad (2.2)$$

As we shall see in Section 2.1.5, this relationship leads to an important characterization of optimal behavior, which finally offers an iterative algorithm for finding optimal decision strategies. To get to this point, it is convenient to introduce a second type of value function, which takes as additional argument the immediate next action of the agent and considers the resulting conditional process, i.e.

$$\begin{aligned} Q^\pi(s, a) &\triangleq \mathbb{E} \left[r_1 + \gamma r_2 + \gamma^2 r_3 + \dots \mid s_1 = s, a_1 = a, \pi \right] \\ &= \mathbb{E} \left[R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \dots \mid s_1 = s, a_1 = a, \pi \right]. \end{aligned} \quad (2.3)$$

To distinguish from the state values in Equation (2.1), the corresponding quantities are commonly referred to as *state-action values*, *action values*, or *Q-values*, which reflect the expected return for executing a particular action at a given state.

2.1.4 Policy Evaluation

To assess the performance of a particular policy in the context of a given SDM problem, we need to compute the corresponding value function in consideration of the chosen performance criterion—a procedure which is known as *policy evaluation*. By imposing a structural condition on the value function, the recursive relationship in Equation (2.2)

provides a computational tool for this task. More specifically, by representing all terms of the infinite sum in Equation (2.1) implicitly, it allows to formulate the problem as a linear fixed-point problem, which can be in fact solved in closed form. To find its solution, it is convenient to rewrite the value recursion in matrix notation, i.e.

$$\mathbf{V}^\pi = \mathbf{R} + \gamma \mathbf{T}^\pi \mathbf{V}^\pi,$$

where $[\mathbf{V}^\pi]_i \triangleq V^\pi(s_i)$, $[\mathbf{T}^\pi]_{i,j} \triangleq \mathcal{T}(s_j | s_i, \pi(s_i))$, and $[\mathbf{R}]_i \triangleq R(s_i)$. The analytic expression for \mathbf{V}^π is then obtained via matrix inversion, i.e.

$$\mathbf{V}^\pi = (\mathbf{I} - \gamma \mathbf{T}^\pi)^{-1} \mathbf{R}, \quad (2.4)$$

where \mathbf{I} represents the identity matrix of size $|\mathcal{S}| \times |\mathcal{S}|$. Unfortunately, as the computational effort for this inversion scales cubically with the size of the state space, a value computation based on Equation (2.4) becomes infeasible for most problems; hence, the following iterative estimation scheme is typically preferred in practice,

$$\hat{\mathbf{V}}^\pi \leftarrow \mathbf{R} + \gamma \mathbf{T}^\pi \hat{\mathbf{V}}^\pi. \quad (2.5)$$

Herein, $\hat{\mathbf{V}}^\pi$ denotes the current estimate of the (vectorized) value function. Since the above operation realizes a contraction mapping on $\hat{\mathbf{V}}^\pi$, the estimate is guaranteed to converge to the unique fixed point \mathbf{V}^π irrespective of the initialization point [SB98].

2.1.5 Bellman's Principle of Optimality

While the policy evaluation procedure in Section 2.1.4 provides a simple way to quantify the performance of an arbitrary policy in a given MDP environment, our primary goal is to find a policy that optimally solves the underlying task. To this end, it is necessary to define first what we mean by the term *optimal*.

Intuitively, a policy should be considered optimal if it achieves the *highest value at all states*. However, in order to approve this definition, we need to ask whether such an object exists in the first place, as it is not directly apparent if one can always find a single maximizing decision rule for all states. The answer is given by one of the fundamental results from the theory of MDPs (see, for example, [Put94]), which guarantees that there exists an optimal (in the above sense) *deterministic Markov* policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ provided that certain mild conditions are fulfilled, i.e.

$$\exists \pi^* : V^{\pi^*}(s) \geq V^\pi(s) \quad \forall \pi, s. \quad (2.6)$$

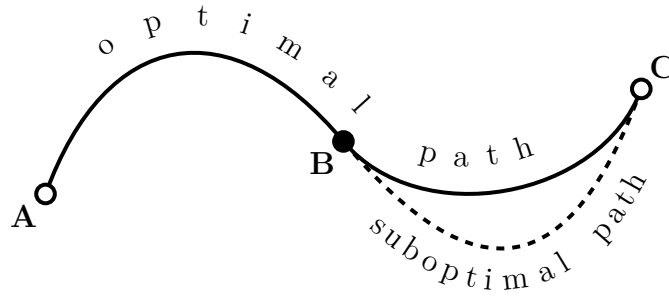


Figure 2.2: Illustration of Bellman’s principle of optimality. If a path from a point **A** to a point **C** is optimal, then also the subpath from any intermediate point **B** on that path must be optimal. In the context of MDPs, the points **A**, **B** and **C** represent states, while a path corresponds to a certain sequence of states/actions, called a *trajectory*.

In other words, optimal policies are indeed optimal *everywhere*. Since an optimal policy, by definition, yields the highest future return in expectation, it must further hold that the assigned actions maximize the corresponding Q-function at all states, i.e.

$$\pi^*(s) \in \arg \max_{a \in \mathcal{A}} Q^{\pi^*}(s, a).$$

The essence of this recursive relation is summarized in Bellman’s famous *principle of optimality*, which is illustrated in Figure 2.2:

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

(Bellman, 1957)

The corresponding *optimal value function* is characterized by the following recursion equation, which is generally known as *Bellman’s optimality equation*,

$$V^{\pi^*}(s) = R(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s' | s, a) V^{\pi^*}(s'). \quad (2.7)$$

For brevity of notation, we denote the optimal value functions by $V^* \triangleq V^{\pi^*}$ and $Q^* \triangleq Q^{\pi^*}$, respectively.

2.1.6 Stochastic Planning

Just like the value recursion in Equation (2.2) characterizes a policy in terms of its expected return, Bellman’s equation provides a characterization of *optimal policies* by

imposing a necessary (and sufficient) optimality condition. However, while the equation can be used to verify if a given policy is optimal, it does not tell how to find such a policy in the first place.

The problem of finding optimal policies in MDPs is known as *stochastic planning*. In the literature, one can find a number of well-established algorithms that allow to compute exact or approximate solutions to the problem. In the following, we focus on the most basic planning algorithm, called *value iteration*, which represents a specific type of dynamic programming [Bel57]. Similar to the iterative scheme described in Equation (2.5), the algorithm is based on an iterative variant of Bellman’s equation and comes with the same convergence guarantees, i.e.

$$\hat{V}^*(s) \leftarrow R(s) + \gamma \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s' | s, a) \hat{V}^*(s'). \quad (2.8)$$

Once the optimal value function V^* has been determined as the unique fixed point, an optimal policy can be found via one-step lookahead planning using Equation (2.6). The required optimal state-action value function is given through the following relationship,

$$Q^\pi(s, a) = R(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s' | s, a) V^\pi(s'), \quad (2.9)$$

which follows directly from Equation (2.3) and holds for any policy including π^* .

Other common planning algorithms include the related policy iteration algorithm, which finds the optimal solution through repeated execution of the policy evaluation procedure in Equation (2.5) alternated with additional policy improvement steps [SB98], and optimization methods based on linear programming [FR03].

2.1.7 Reinforcement Learning

As the word *planning* suggests, the solution methods described in Section 2.1.6 address the SDM problem by effectively taking into account all possible future evolutions of a system and choosing the best actions accordingly. In fact, the recursion equations in Sections 2.1.4 and 2.1.5 can be interpreted as extrapolations of the system state that “anticipate” the consequences of executing a particular policy. This consideration of future events is only possible if the planning method has access to the transition model \mathcal{T} and the reward function R , which capture the dynamics of the environment and the underlying task description.

Unfortunately, the exact dynamics of a system are often unknown or not fully specified, and the agent might be unclear about the specific reward mechanism of the environment. This can have several reasons; for example, it could happen that the agent encounters an unexpected situation during the decision-making process or, if an unknown terrain is entered for the first time, the agent could simply have no prior knowledge of the underlying system mechanics. Unfortunately, if the system environment is unknown, planning-based solution methods cannot be applied. Moreover, it has been shown that the use of inaccurate system models can result in poor performance when the learned behavior is executed in the real environment [AS97a; AS97b].

An alternative solution paradigm, which specifically addresses these problem, is *reinforcement learning (RL)*. The basic principle of RL is to learn a suitable behavioral policy from experience, by *interacting* with the environment, rather than relying solely on model-based planning. Hence, unlike the previously discussed methods, RL offers a *data-driven* approach to sequential decision making.

The field of RL methods is vast and a detailed discussion of existing approaches would go beyond the scope of this introduction. To convey the basic principle of experience-based learning, we thus focus on a specific learning algorithm, i.e., *Q-learning* [WD92], which represents one of the milestones in RL and forms the basis for one of the learning algorithms presented in this thesis. For a general overview of the field, we refer to the following books and survey papers [SB98; WO12; KLM96; KBP13; DNP13], which cover the three basic RL methodologies: (i) model-based RL, which uses experience-based models of the environment for approximate planning, (ii) model-free RL, which estimates value functions directly from experience data, and (iii) policy search methods, which perform the optimization directly in the policy space.

Q-Learning

Q-learning is a model-free off-policy RL algorithm that has received a lot of attention in past decades. The description *model-free* refers to the fact that the algorithm does not build an explicit model of the environment to solve Bellman’s equation. The term *off-policy* indicates that the exploration policy, which is used to gather new experience during execution of the algorithm, can be selected without consideration of the task encoded by the reward function. While the original algorithm is designed for finite MDP environments, many adapted versions of Q-learning exist (for example, LSTDQ [LP03] or DQN [Mni+15]), which extend its capabilities to continuous domains by using function approximation techniques.

The basic Q-learning principle is based on an iterative refinement of Q-value estimates, which is achieved using a convex combination of the current estimates and those computed from new incoming data, i.e.

$$\hat{Q}^*(s_t, a_t) \leftarrow (1 - \alpha_t) \hat{Q}^*(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_{a' \in \mathcal{A}} \hat{Q}^*(s_{t+1}, a_{t+1}) \right), \quad (2.10)$$

where $\alpha_t \in (0, 1)$ denotes the learning rate of the algorithm at the t th iteration. The experiences $\{(s_t, a_t, r_t, s_{t+1})\}$ are obtained from the data stream that is generated by interacting with the system environment, according to the generative process described in Section 2.1.2.

The second part of the update expression in Equation (2.10) can be interpreted as an instantaneous estimate of the future return, approximating the recursive relationship

$$Q^*(s, a) = R(s) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s' | s, a)} \left[\max_{a' \in \mathcal{A}} Q^*(s', a') \right],$$

which follows from the definition of the Q-function in Equation (2.3) and the value recursion in Equation (2.2). Yet, a perhaps more intuitive view on the update is given by the following representation, which reveals the urge to satisfy Bellman's optimality condition by compensating for the *temporal difference error* between the current estimate of the Q-value and the corresponding one-step lookahead prediction, i.e.

$$\hat{Q}^*(s_t, a_t) \leftarrow \hat{Q}^*(s_t, a_t) + \alpha_t \underbrace{\left(\overbrace{r_t + \gamma \max_{a' \in \mathcal{A}} \hat{Q}^*(s_{t+1}, a_{t+1})}^{\text{one-step lookahead prediction}} - \overbrace{\hat{Q}^*(s_t, a_t)}^{\text{current estimate}} \right)}_{\text{temporal difference error}}.$$

Herein, the error acts as a correction term that incorporates the incoming information about the reward function/the system dynamics that is contained in the newly experienced transition (s_t, a_t, r_t, s_{t+1}) . Provided that all state-action combinations are encountered infinitely often and that the learning rate satisfies the stochastic approximation conditions, i.e.

$$\sum_{t=0}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty,$$

the estimate \hat{Q}^* is guaranteed to converge to the optimal state-action function Q^* .

2.2 Learning from Demonstration

Reinforcement learning is a powerful instrument for solving SDM problems that has shown impressive results, for example, in multi-agent coordination [BBD08], game playing [Tes95; Sil+16], and system control based on low-level input [Mni+15]. However, dealing with complex or interacting systems whose (joint) state space is large remains a challenge. For such systems, the exploration of the state space typically becomes problematic and off-the-shelf methods often fail at finding control policies that efficiently exploit the system dynamics.

Struggling with these problems in RL on the one hand, we regularly observe optimized behavioral strategies that are able to cope with complex, high-dimensional problems on the other hand. Such situations are encountered, for example, in problems that require full-body control, such as biped locomotion, which is an easy task for humans and other living beings but poses a challenging control problem in robotics [BF97; PVS03]. Other fascinating examples of specialized strategies are found in nature, like the seemingly instantaneous collective reaction of animal herds in the event of predator attacks [VZ12]. In such cases, where observational data of an expert behavior is available, *learning from demonstration (LfD)* [Arg+09; Sch99] provides a complementary solution paradigm to classical RL that offers a promising approach to both system identification and behavior optimization.

In contrast to RL, where experience is gathered solely through interaction with an environment, LfD methods (additionally) exploit domain-specific knowledge that is provided by an expert demonstrator. This allows LfD-based methods to focus on the relevant parts of a system’s state space [Arg+09] and to avoid the need of tedious exploration steps performed by black box RL methods, which often require an impractically high number of interactions with the environment [DFR15] and always come with the risk of letting the system run into undesired or unsafe states [AN05]. Interestingly, several studies showed that LfD-build strategies can even outperform the expert by filtering out suboptimal decisions from the demonstration set [Sam+92; ACN10].

Historically, most research in the field has been driven by the robotics community (see [Arg+09] for an overview), which is explained by the fact that LfD models offer a flexible interface for robot skill acquisition [Dil+00]. Nonetheless, the success of LfD has also triggered developments in other research areas, such as human-machine interaction [Pie13], cognitive science [RD11] and behavioral psychology [Rot08]. While all LfD methods basically pursue the same goal—that is, constructing behavioral models based on demon-

stration data—the number of possible views on the learning problem is huge, and consequently, many different ideas and concepts have been presented in the past. Depending on the particular objective and assumptions made about the expert behavior, the problem is sometimes referred to as apprenticeship learning [AN04], imitation learning [Sch99], behavioral cloning [Sam+92], inverse decision making [JLK11], inverse reinforcement learning [NR00], inverse optimal control [DT10], plan recognition [CG93], policy recognition [ŠZK18a], preference elicitation [RD11], programming by demonstration [Bil+08], learning by watching [KII94], or teaching by showing [Miy+96]. Note that the boundaries between these problem domains are fluid and many methods rely on similar principles, like intention extraction [ZJ12; NR00; HZ15], trajectory matching [Eng+13; Mae+14], or action reconstruction via supervised learning [Pom91; AS97b; SS10].

Though there are many taxonomies to classify LfD approaches, a fundamental distinction can be made between those that model the demonstrated behavior on the intentional level and such that work on the decision level. Adopting the terminology in [PG17], we refer to these two types as *intentional models* and *subintentional models*. The following sections provide a basic overview of both paradigms.

2.2.1 Subintentional Modeling

Subintentional LfD methods aim at capturing the behavior of an agent on the decision level, by constructing policy models that mimic the decision-making strategy of the agent. In contrast to intentional approaches (Section 2.2.2), they do not ask about the underlying motives for executing a particular strategy but focus only on its reconstruction.

The difficulty of the reconstruction problem herein depends on various factors. One of them is the perception of the demonstration data, which determines whether state and action information of the demonstrator is accessible directly or has to be inferred from other measurements. For example, in robotics, demonstrations can be acquired by watching the demonstrator [KII94] (exteroceptive sensing) or via kinesthetic teaching [KCC10]/teleoperation [ACN10] (proprioceptive sensing); however, only in the latter case a complete record of the state and action history is available.

Also, the final problem formulation depends on our knowledge of the system parameters (i.e., the action space and transition dynamics) and our prior assumptions about the demonstrated behavior, for example, on whether we believe that the expert follows a deterministic policy or executes a stochastic behavior. In the simplest scenario, where the policy is deterministic and demonstrations are provided in form of state-action pairs

$\mathcal{D} = \{s_d, a_d\}_{d=1}^D$, the modeling scenario reduces to a conventional supervised learning setting and the goal becomes to recover the underlying functional mapping $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that describes the action selection mechanism of the agent. In Chapter 3, we begin with a slightly more complicated version of this problem, where we have only noisy observations of the states visited by the expert, without having access to the corresponding action sequence. The reconstruction problem becomes significantly more challenging once we move over to stochastic policies in Chapter 4, which requires a probabilistic modeling of the involved action variables and their correlations across system states.

2.2.2 Intentional Modeling

In contrast to subintentional modeling, which is only concerned with the *consequences* of an agent’s intention (i.e., the executed actions), intentional modeling aims at understanding the agent’s decision-making process as a whole, by reconstruction the causal relationships of the demonstrated behavioral patterns. Since this provides complete insight into the motives behind an action, intentional modeling is sometimes referred to as *true imitation* [BS02].

Because intentional models try to “invert” the decision-making process of an agent, they always involve some model of rationality that *explains* the agent’s choice of action in a given situation. In many cases, this model takes the form of an MDP, which is usually supplemented by additional assumptions about the agent behavior that make the inference problem solvable (see next paragraph). In the following, we focus on a particular modeling principle — inverse reinforcement learning (IRL) — which forms basis for the contributions presented in Parts II and III.

Inverse Reinforcement Learning

While the problem of LfD has already been studied since the 1980s [AS97b], the idea of IRL has become popular in the last two decades. In their seminal paper, Ng and Russell [NR00] formally defined the IRL problem as the problem of finding the reward function that is being optimized by an observed expert demonstrator. Based on the properties of optimal policies in MDPs, the authors showed that the set of potential reward functions that make a given policy $\pi \triangleq a_1$ optimal is characterized by the following vector inequality,

$$(\mathbf{T}^{a_1} - \mathbf{T}^a)(\mathbf{I} - \gamma \mathbf{T}^{a_1})^{-1} \mathbf{R} \succeq \mathbf{0} \quad \forall a \in \mathcal{A} \setminus a_1, \quad (2.11)$$

where \succeq indicates element-wise inequality, $[\mathbf{T}^a]_{i,j} \triangleq \mathcal{T}(s_j | s_i, a)$, and $\mathbf{0}$ is the zero vector. Note that any deterministic policy can be written as $\pi = a_1$ by renaming actions accordingly.

Unfortunately, the inequality in (2.11) provides only a necessary optimality condition for the reward function, which leads to an ill-defined formulation of the IRL problem. The reason for this is that the relationship between intentions and behavior is generally ambiguous, since tasks can be often solved in several ways and different intentions can lead to the same behavioral strategy. In the context of MDPs, this means that there generally exists no one-to-one correspondence between reward functions and policies. In fact, for a given MDP model, there can be infinitely many optimal policies and, vice versa, there is an infinite number of reward functions that make a given policy optimal ($\mathbf{R} = \mathbf{0}$ being one of them). Because of this property, which is inherent to the IRL problem, additional assumptions are required in order to determine a unique solution. In the past, researchers have proposed various strategies to tackle the problem, e.g., by reducing the class of MDP models (linearly solvable MDPs [DT10]), introducing additional objectives (max-margin IRL [NR00] and max-entropy IRL [Zie+08]), or using probabilistic reasoning (Bayesian IRL [RA07]). A review of some basic techniques can be found in [ZJ12].

A second, more practical issue arising in the estimation problem is that the expert demonstrations might not be optimal in the first place, e.g., because of noise in the data generation/acquisition process or due to uncertainties by the expert regarding the task. More recent methods account for this problem by producing conservative policies from the estimated reward function [SS08] or by explicitly modeling the degree of rationality of the expert’s decisions [RA07; LPK11]. However, the impact of the underlying rationality model on the extracted reward function has not been studied. In Part II, we investigate this problem in detail and show that the way suboptimality is addressed by the rationality model can have significant effects on the estimation result.

2.2.3 Intentional Modeling versus Subintentional Modeling

Learning from demonstration has become a viable alternative to classical RL, and both intentional and subintentional methods have progressed rapidly in the recent past. However, each strategy comes with its own advantages and disadvantages, and there is no general answer to which of the two paradigms offers the better modeling approach—partly because most existing works on LfD focus on either on the two research directions (see [PGP13; PGP17] for notable exceptions).

On the one hand, intentional models provide richer descriptions of an agent’s behavior. This offers the possibility to analyze the preferences of the demonstrator, which makes these models interesting from a psychological point of view [RD11]. For the same reason, intentional models are said to have better generalization abilities, the underlying rationale being that an agent’s intention provides the most robust, succinct and transferable description of a task [NR00]. On the other hand, the learning objective of intentional models is inherently ill-defined, which requires additional assumptions about the expert that can bias the reconstruction of the behavior. In fact, if the intention of the agent is misunderstood, any subsequent generalization attempt will fail trivially. Also, intentional models consider the LfD problem at a complexity level that is not necessarily required in all modeling scenarios, e.g., when the goal is to analyze a certain local behavioral pattern.

For a detailed comparison of the advantages and disadvantages of both modeling paradigms, we point to the comprehensive discussion in [PGP13].

PART I

Single-agent Systems: Subintentional Modeling

3	Policy Recognition: A Simple Approach	29
4	Parametric Policy Recognition	38
5	Nonparametric Policy Recognition	54
6	Experimental Results	60
7	Summary	69

In this first part of the thesis, we focus on single-agent modeling and address the problem of reconstructing the decision-making strategy of an observed demonstrator from incomplete measurements of the underlying sequential process. Herein, we consider the LfD problem in a subintentional context and model the demonstrated behavior directly at the action level, without reasoning about the latent intentions of the agent. This means in particular that we do not address the question if, and in what sense, the shown behavior is optimal. Instead, our goal is to identify the executed policy based on the provided behavior data—a problem we refer to as *policy recognition*.

Before we get to the full formulation of this task in Chapter 4, we start with a simpler version of the problem to provide a basic understanding of the difficulties involved.

3

Policy Recognition: A Simple Approach

In the following, we consider a reduced MDP of the form $(\mathcal{S}, \mathcal{A}, \mathcal{T})$ without reward function. In the literature, this model is sometimes referred to as a controlled Markov process (CMP) [DR11; RD11] or MDP\textbackslash R [AN04; SS08; MH12], to emphasize the

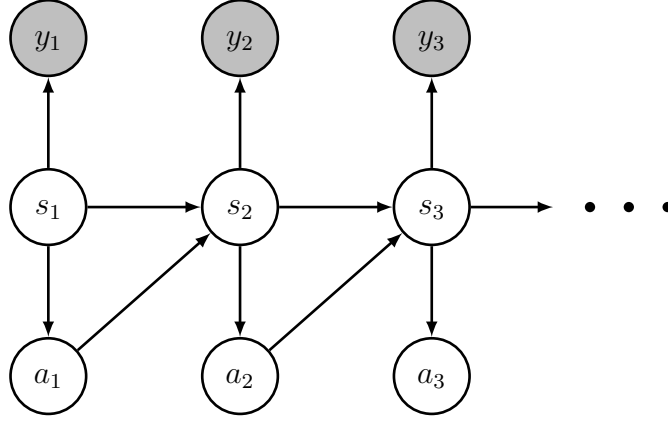


Figure 3.1: Bayesian network of the simplified policy recognition model. The model has the same structure as the MDP shown in Figure 2.1 but the rewards are replaced with noisy state observations (shaded nodes).

nonexistence of a reward mechanism. For the moment, we assume that both state and action space are finite in size, and we represent their elements by integer values.

Suppose we observe an agent executing a deterministic policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ in this environment. More specifically, suppose that we have access to a noisy version $\mathbf{y} \triangleq (y_1, \dots, y_T) \in \mathcal{Y}^T$ of the agent's state trajectory $\mathbf{s} \triangleq (s_1, \dots, s_T) \in \mathcal{S}^T$ that was generated under π according to the stochastic process described in Section 2.1.2. Herein, T denotes the (observation) trajectory length and \mathcal{Y} is a finite set of possible state observations. The policy recognition problem can then be formulated as the task of estimating the agent's policy π based on the noisy observation sequence \mathbf{y} .

Working in a finite state space, we can express the policy as a collection of action assignments, i.e., $\boldsymbol{\pi} \triangleq (\pi_1, \dots, \pi_{|\mathcal{S}|}) \in \mathcal{A}^{|\mathcal{S}|}$, where π_i represents the action choice of the agent at state i . Accordingly, the joint distribution of true states \mathbf{s} and observations \mathbf{y} can be written as

$$p(\mathbf{s}, \mathbf{y} \mid \boldsymbol{\pi}) = p_1(s_1) \prod_{t=1}^{T-1} \mathcal{T}(s_{t+1} \mid s_t, \pi_{s_t}) \prod_{t=1}^T p(y_t \mid s_t), \quad (3.1)$$

where p_1 represents the initial state distribution of the agent. The structure of that joint distribution is visualized in Figure 3.1, which illustrates the statistical relationships between the involved variables. Note that the model can be generalized straightforwardly to multiple trajectories, as they are conditionally independent given $\boldsymbol{\pi}$.

3.1 Maximum Likelihood Estimation

As a first attempt, we approach the policy recognition problem via maximum likelihood (ML) estimation, i.e.

$$\hat{\boldsymbol{\pi}}^{\text{ML}} \triangleq \arg \max_{\boldsymbol{\pi} \in \mathcal{A}^{|\mathcal{S}|}} p(\mathbf{y} | \boldsymbol{\pi}). \quad (3.2)$$

Unfortunately, the ML estimate cannot be computed in closed form, because the agent’s true state sequence \mathbf{s} is unknown. However, an approximate solution can be found in an iterative way using the expectation maximization (EM) algorithm [DLR77]. To this end, we first compute the expectation of the complete log-likelihood function according to Equation (3.1) for an initial guess $\boldsymbol{\pi}'$ of the agent’s policy (E-step), i.e.

$$\begin{aligned} \mathcal{Q}^{\text{ML}}(\boldsymbol{\pi}, \boldsymbol{\pi}') &\triangleq \sum_{\mathbf{s} \in \mathcal{S}^T} p(\mathbf{s} | \mathbf{y}, \boldsymbol{\pi}') \log p(\mathbf{s}, \mathbf{y} | \boldsymbol{\pi}) \\ &\stackrel{\text{c}}{=} \sum_{t=1}^{T-1} \sum_{s_{t+1} \in \mathcal{S}} \sum_{s_t \in \mathcal{S}} p(s_{t+1}, s_t | \mathbf{y}, \boldsymbol{\pi}') \log p(s_{t+1} | s_t, \pi_{s_t}), \end{aligned} \quad (3.3)$$

where $\stackrel{\text{c}}{=}$ indicates equality up to an additive constant. Maximizing $\mathcal{Q}^{\text{ML}}(\boldsymbol{\pi}, \boldsymbol{\pi}')$ with respect to $\boldsymbol{\pi}$ (M-step) is guaranteed to monotonically increase the marginal likelihood $p(\mathbf{y} | \boldsymbol{\pi})$, so that the sequence of estimates obtained by iterating between both steps converges to a local maximum. Note that the distribution $p(s_{t+1}, s_t | \mathbf{y}, \boldsymbol{\pi}')$ in Equation (3.3) can be efficiently computed for all time instants $t \in \{1, \dots, T-1\}$ using the Baum-Welch algorithm [Bau+70].

By reordering summations in Equation (3.3), we can separate the individual contributions of all policy parameters in the E-step, i.e.

$$\begin{aligned} \mathcal{Q}^{\text{ML}}(\boldsymbol{\pi}, \boldsymbol{\pi}') &= \sum_{i=1}^{|\mathcal{S}|} \mathcal{Q}_i^{\text{ML}}(\pi_i, \boldsymbol{\pi}'), \end{aligned} \quad (3.4)$$

where $\mathcal{Q}_i^{\text{ML}}(\pi_i, \boldsymbol{\pi}') \triangleq \sum_{t=1}^{T-1} \sum_{s_{t+1} \in \mathcal{S}} p(s_{t+1}, s_t = i | \mathbf{y}, \boldsymbol{\pi}') \log p(s_{t+1} | s_t = i, \pi_i),$

which shows that the objective function decouples into distinct terms $\{\mathcal{Q}_i^{\text{ML}}(\pi_i, \boldsymbol{\pi}')\}_{i=1}^{|\mathcal{S}|}$ that can be optimized independently. Accordingly, the M-step can be performed separately for each state in \mathcal{S} .

However, the decomposition in Equation (3.4) also shows that the ML approach does not provide solutions for states that are “far” from the given demonstrations. This can be seen from the following argument: if $p(s_t = i | \mathbf{y})$ is zero — meaning the hypothesis

that the agent visited state i at time t is not compatible with our measurements \mathbf{y} under the assumed observation model— then also $p(s_{t+1}, s_t = i | \mathbf{y}, \boldsymbol{\pi}')$ is zero and so is the corresponding term Q_i^{ML} , *irrespective of the assignment of π_i* . In other words, the ML solution cannot generalize the observed behavior to regions of the states space that were not visited by the agent. This result is not surprising, as no statistical assumptions about the agent’s action assignments were made.

3.2 Maximum A Posteriori Estimation

Since the ML approach can solve the policy recognition problem only partially, we now present an alternative estimation strategy that incorporates additional information about the agent behavior in form of a *policy prior model*. To motivate the approach, consider the policies shown in Figure 3.2, which optimally solve four of the test scenarios described in Section 3.3. What becomes immediately apparent is that the policies are highly structured and show a significant degree of spatial correlation between neighboring states.

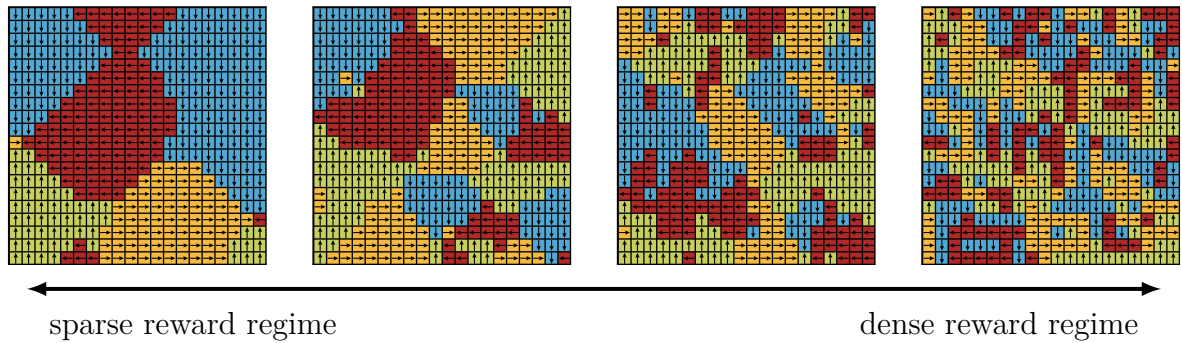


Figure 3.2: Optimal policies in the Gridworld domain (Section 3.3) for different reward regimes. All examples show a significant degree of spatial correlation.

The reason for this phenomenon can be traced back to the transition dynamics of the system, which induce a spatial relationship between states that causes “nearby” states to share similar action assignments. This relationship can be understood intuitively if we interpret the system states as physical locations and actions as motion commands in certain directions. In order to reach a particular goal state in the environment, the agent needs to execute a specific sequence of actions that is both correlated in space and time. A formal definition of this relationship will have to wait until Part II (see Section 10.2.1), where we talk about goal-oriented behavior. For now, we simply use these insights to improve our estimation strategy.

In the following, our goal is to exploit the underlying structure of the state space in order to establish a link between visited parts of the space and regions where no measurements were obtained. For this purpose, we describe the agent’s policy using a Potts model [Pot52], which represents a specific type of Markov random field (MRF) based on pairwise clique potentials [KF09]. The model is defined by the following relation,

$$p(\boldsymbol{\pi}) \propto \prod_{i=1}^{|S|} \exp \left(\frac{\beta}{2} \sum_{j \in \mathcal{N}_i} J_{\pi_i, \pi_j} \delta(\pi_i, \pi_j) \right), \quad (3.5)$$

where δ represents Kronecker’s delta function, $\beta \in [0, \infty)$ is the (inverse) temperature of the model controlling the coupling strength of the policy parameters, and $J_{m,n} \in [0, \infty)$ encodes the “similarity” of action m and action n . Since this relationship is naturally undirected, the elements $\{J_{m,n}\}$ are arranged in a symmetric matrix \mathbf{J} . The neighborhood \mathcal{N}_i is used to describe the range of the spatial influence of parameter π_i on the remaining policy parameters, which are in the following summarized as $\boldsymbol{\pi}_{\setminus i}$. The neighborhood relationship is assumed to be symmetric, too, meaning that state i is automatically a neighbor of state j whenever state j is a neighbor of state i .

With the new prior model at hand, we seek for the maximum a posteriori (MAP) estimate of $\boldsymbol{\pi}$, i.e.

$$\hat{\boldsymbol{\pi}}^{\text{MAP}} = \arg \max_{\boldsymbol{\pi} \in \mathcal{A}^{|S|}} p(\boldsymbol{\pi} | \mathbf{y}) = \arg \max_{\boldsymbol{\pi} \in \mathcal{A}^{|S|}} \left(\log p(\mathbf{y} | \boldsymbol{\pi}) + \log p(\boldsymbol{\pi}) \right).$$

As in Section 3.1, the estimate can be computed via EM, by augmenting the objective function in the E-step (Equation 3.3) with an additional prior term [DLR77], i.e.

$$\mathcal{Q}^{\text{MAP}}(\boldsymbol{\pi}, \boldsymbol{\pi}') = \mathcal{Q}^{\text{ML}}(\boldsymbol{\pi}, \boldsymbol{\pi}') + \log p(\boldsymbol{\pi}). \quad (3.6)$$

Unfortunately, the corresponding M-step becomes infeasible under the MRF prior because the function no longer decouples into individual terms, requiring an optimization over an exponentially large space. Yet, we can arrive at a local maximum by applying the iterated conditional modes (ICM) algorithm [Bes86], which optimizes one parameter at a time. As in the ML case, we define a local objective function $\mathcal{Q}_i^{\text{MAP}}$ for each parameter π_i . This time, however, its value additionally depends on the assignment of the remaining parameters in the neighborhood \mathcal{N}_i through the policy prior model, i.e.

$$\begin{aligned} \mathcal{Q}_i^{\text{MAP}}(\pi_i, \boldsymbol{\pi}_{\setminus i}, \boldsymbol{\pi}') &= \mathcal{Q}_i^{\text{ML}}(\boldsymbol{\pi}, \boldsymbol{\pi}') + \log p(\pi_i | \boldsymbol{\pi}_{\setminus i}) \\ &\stackrel{c}{=} \mathcal{Q}_i^{\text{ML}}(\boldsymbol{\pi}, \boldsymbol{\pi}') + \beta \sum_{j \in \mathcal{N}_i} J_{\pi_i, \pi_j} \delta(\pi_i, \pi_j). \end{aligned} \quad (3.7)$$

Note that the scaling factor $\frac{1}{2}$ from Equation (3.5) has vanished due to the symmetry properties of \mathcal{N}_i and \mathbf{J} . Optimizing the local functions $\{\mathcal{Q}_i^{\text{MAP}}\}$ one after another with respect to their individual policy parameters will monotonically increase the value of the global object in Equation (3.6), which allows us to iteratively refine our estimate.

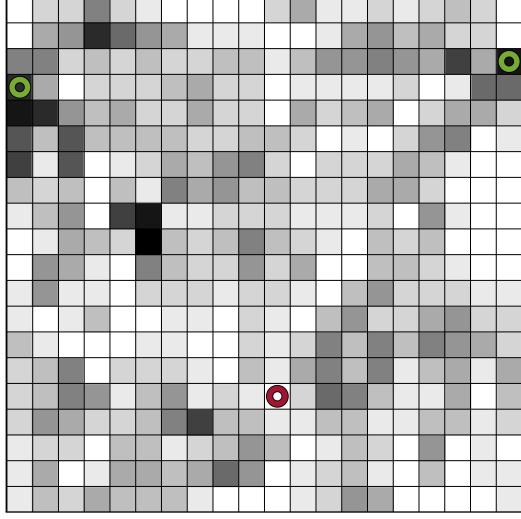
3.3 Experimental Results

To compare both estimation approaches, we consider a set of randomly generated test scenarios, each consisting of $20 \times 20 = 400$ distinct states arranged on a regular grid. The setting corresponds to a particular instance of the common Gridworld benchmark used in RL [SB98]. An example scenario is depicted in Figure 3.3.

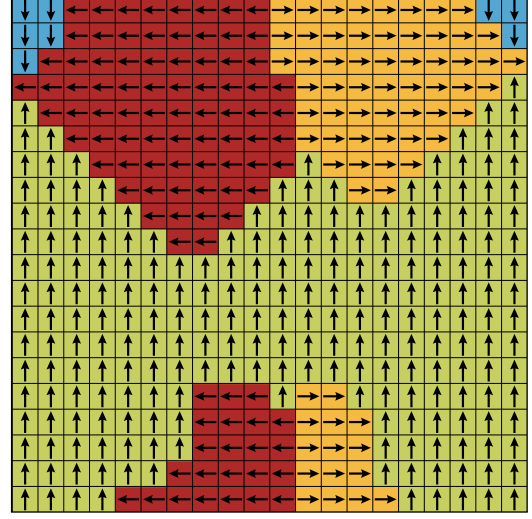
The transition dynamics of the environment are defined as follows: an agent living in the Gridworld can choose among four actions, representing the four cardinal directions *north*, *east*, *south* and *west*. By executing one of these action, the agent moves in the desired direction with a probability of 0.6. With the remaining probability of 0.4, the agent accidentally moves in one of the other three directions or stays in place. Whenever the resulting move would let the agent hit the boundary of the world, the position of the agent remains unaltered. At any point in time we observe the agent’s true position with a probability of 0.6. Otherwise, we mistakenly measure its position at one the neighboring four states (where all probability mass “lying outside” the world is again shifted to the agent’s true position).

In order to define a task for the agent, each state is assigned a fixed reward with a given probability $\tau \in (0, 1]$, where the reward values are drawn from a standard normal distribution. Worlds that contain no rewards are discarded. The policy of the agent is given by the optimal solution to the underlying MDP, where we assume a discount factor of $\gamma = 0.9$. The initial state distribution p_1 is chosen as the uniform distribution on \mathcal{S} .

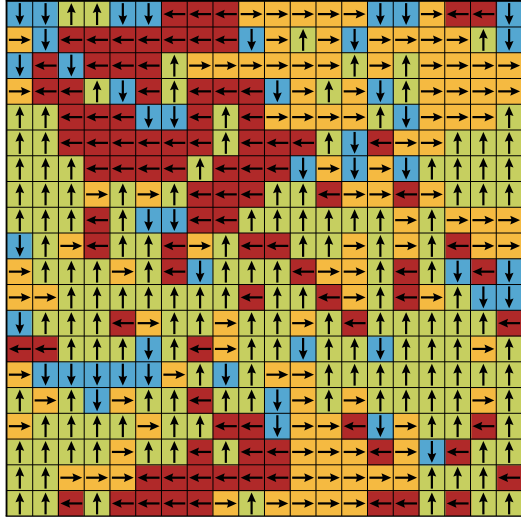
Correlation Structure First, we investigate the correlation structure of the induced policies models, to empirically verify the spatial smoothness assumption that motivated the use of the Potts model in Section 3.2. Figure 3.4 shows the estimated probabilities for the three possible action constellations that can occur at neighboring states, which we obtained from randomly generated test scenarios using different reward probabilities τ . As expected, we observe a high degree of correlation between the local action assignments, which is preserved even in dense reward regimes.



(a) visitation frequency & rewards



(b) ground truth



(c) ML estimate

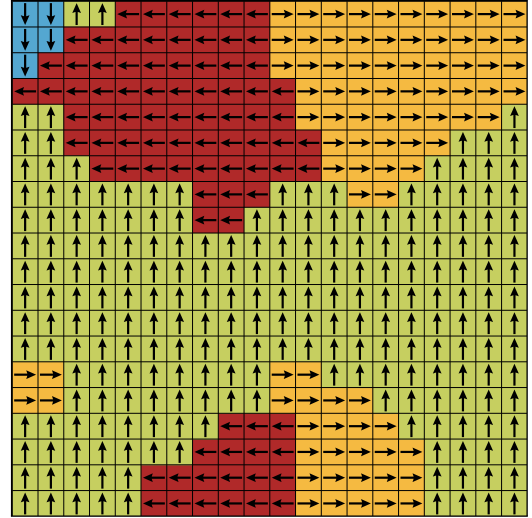

 (d) MAP estimate ($\beta = 2$)

Figure 3.3: Comparison of the ML and MAP estimation approaches on a randomly generated Gridworld scenario for $\tau = 0.01$. Both estimates are based on 200 expert trajectories, each comprising four state transitions. Dark regions in subfigure (a) indicate a high state visitation frequency by the expert. The two green circles indicate positive rewards, the red circle represents negative reward.

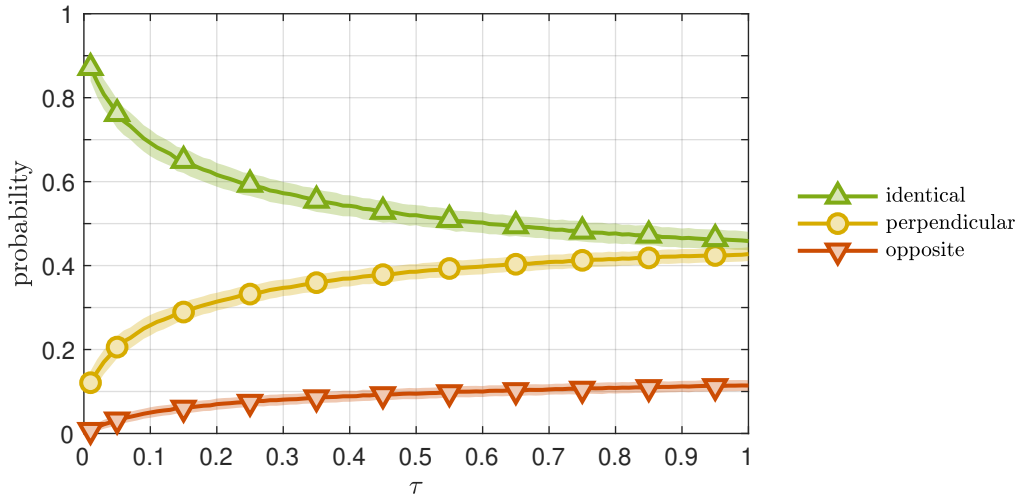


Figure 3.4: Spatial correlation of the optimal policies in the Gridworld scenario for different reward probabilities τ . The graphs depict the estimated probabilities that two adjacent states are assigned actions that point to identical, perpendicular, or opposite directions. Shown are the empirical mean values and standard deviations obtained from 1000 Monte Carlo runs.

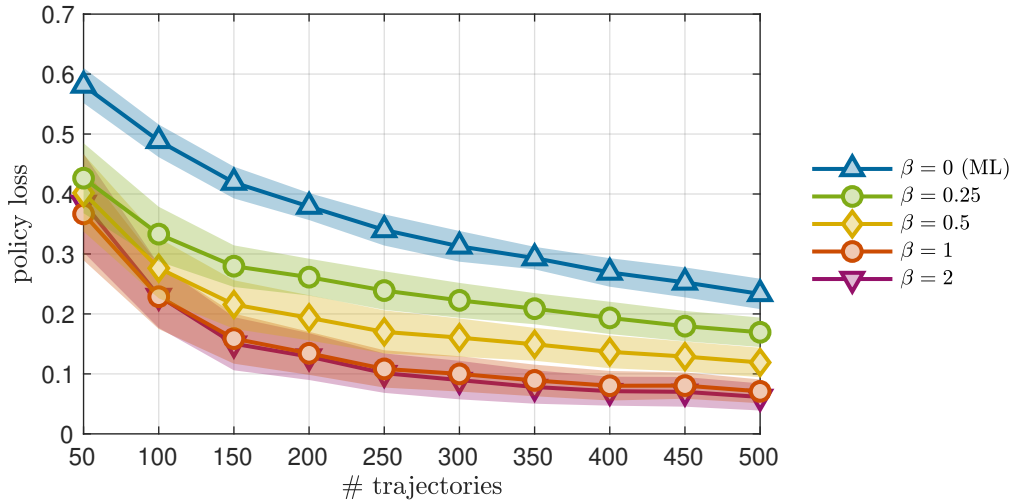


Figure 3.5: Estimated policy loss in the Gridworld scenario over the number of demonstrated trajectories for different prior strengths β . Each trajectory consists of four state transitions. Shown are the empirical mean values and standard deviations obtained from 100 Monte Carlo runs.

Estimation Accuracy Next, we compare the estimation accuracies of the ML and MAP approach in terms of the obtained *policy loss*, which measures the mismatch between the ground truth policy π and our reconstruction $\hat{\pi}$, i.e.

$$L(\pi, \hat{\pi}) \triangleq \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \delta(\pi_i, \hat{\pi}_i).$$

For the evaluation, we consider a sparse reward setting of $\tau = 0.01$. The influence of the reward density on the estimation accuracy is investigated later in Section 11.2. As we saw in Figure 3.4, the local correlation structure in this regime is dominated by neighboring states with identical action assignments. For this reason, we choose the similarity matrix \mathbf{J} as the identity matrix, ignoring all other dependencies between different actions. In order to demonstrate that already a crude prior model can significantly improve the accuracy of the reconstruction, we adopt a simple four-state neighborhood structure in which two states are neighbors if their Manhattan distance on the grid is 1. Figure 3.5 depicts the policy loss over the number of expert trajectories (each comprising four state transitions) for different prior strengths β . The result shows that the MAP approach clearly outperforms the ML solution, with an average loss reduction of about 65% caused by the MRF prior.

3.4 The Next Steps

The simple scenario presented in this chapter is exemplary to highlight one of the main challenges in learning from demonstration data, namely the generalization of observed behavioral patterns based on a finite number of measurements. Since the set of available demonstrations is typically small compared to the size of the system’s state space, a systematic use of prior information is key to successfully solving LfD problems. The results from Section 3.3 show empirically that a pure ML approach is unlikely to produce a reasonable behavioral model even in the simplistic setting where the expert executes a deterministic behavior on a finite state space. However, by exploiting the correlation structure of the expert policy, we were able to construct an adequate representation of the shown behavior.

In the following sections, we develop this basic idea into a more principled inference framework that can be applied to general behavior types. Starting from the previous setting, we gradually reduce our modeling assumptions until we end up with a fully probabilistic policy recognition model that is able to pick up expert behaviors of—in theory—arbitrary complexity.

4

Parametric Policy Recognition

To address the policy recognition problem in its most general form, we will now drop the assumption of observing a deterministic decision-making strategy and study arbitrary stochastic behavioral patterns. Also, we will leave the expectation maximization framework from Chapter 3 and switch to a full Bayesian formulation of the inference problem, with the goal of learning a predictive behavioral model that considers the complete posterior distribution of possible expert strategies.

In order to focus on the key challenges of this task, we assume from here on to have access to noise-free observations of the expert’s states $\mathbf{s} = (s_1, s_2, \dots, s_T)$, which allows us to ignore the additional observation layer in Figure 3.1. This simplifying assumption is easy to justify since working with noisy observations poses no additional challenge from the perspective of behavioral modeling: as exemplified in Chapter 3, the assumption of noisy trajectory data only implies that we need to work with conditional distributions over expert states (see, for example, Equation 3.3). While this requires us to solve an additional smoothing problem [Sär13], the computation of the conditional distribution can be regarded as an inner loop of the estimation problem that integrates seamlessly into the inference procedures presented in the following chapters.

Therefore, our task can be summarized as follows: given a trajectory $\mathbf{s} \in \mathcal{S}^T$ of states visited by the demonstrator, we want to find the predictive action distribution $p(a \mid s^*, \mathbf{s})$ that describes the action selection strategy of the demonstrator at an arbitrary query state $s^* \in \mathcal{S}$. In this chapter, we approach the problem using a parametric policy model, assuming that the expert behavior can be accurately described in terms of a latent parameter $\boldsymbol{\omega} \in \Omega$, which we call the *global control parameter* of the system. Herein, the set Ω describes the parameter space of the policy, which specifies the class of feasible behavioral models. The specific form of Ω will be discussed later. In the following, we write $\pi(a \mid s, \boldsymbol{\omega})$, $\pi : \mathcal{A} \times \mathcal{S} \times \Omega \rightarrow [0, 1]$, to denote the expert’s *local policy* (i.e., the distribution of actions a played by the expert) at any given state s under $\boldsymbol{\omega}$.*

Using a parametric representation for π is convenient as it shifts the recognition task

* Note that the policy model from Chapter 3 is obtained as a special case if the global control parameter $\boldsymbol{\omega}$ is given as a vector of action assignments $(\pi_1, \dots, \pi_{|\mathcal{S}|})$ and all action probability mass at state i is placed on the corresponding assignment π_i , i.e., when $\pi(a = j \mid s = i, \boldsymbol{\omega}) = \delta(j, \pi_i)$.

from determining the possibly infinite set of local policies at all system states \mathcal{S} to inferring the posterior distribution $p(\boldsymbol{\omega} | \mathbf{s})$, which contains all information that is relevant for predicting the expert behavior, i.e.

$$p(a | s^*, \mathbf{s}) = \int_{\Omega} \pi(a | s^*, \boldsymbol{\omega}) p(\boldsymbol{\omega} | \mathbf{s}) d\boldsymbol{\omega}. \quad (4.1)$$

Since the resulting predictive models $\{p(a | s^*, \mathbf{s})\}_{s^* \in \mathcal{S}}$ are coupled through the global control parameter $\boldsymbol{\omega}$ as indicated by the above integral equation, inferring $\boldsymbol{\omega}$ means not only to determine the individual local policies of the expert but also their spatial dependencies. Consequently, learning the structure of $\boldsymbol{\omega}$ from demonstration data can be also interpreted as learning a suitable state representation for the task performed by the expert. This relationship will be discussed in detail in the forthcoming sections. In Chapter 5, we further extend our reasoning to a family of nonparametric models, whose hypothesis class finally covers all stochastic policies on \mathcal{S} .

4.1 Finite State Spaces: The Static Model

First, let us reconsider the scenario from Chapter 3, where we assumed that the expert system can be modeled on a finite state space \mathcal{S} . As before, we denote the cardinality of that space by $|\mathcal{S}|$ and represent its elements by integer values.

Starting with the most general case, we assume that the expert executes an individual control strategy at each system state. Accordingly, we introduce a set of *local control parameters* or *local controllers* $\{\boldsymbol{\theta}_i\}_{i=1}^{|\mathcal{S}|}$, by which we describe the expert's choice of actions. More specifically, we model the executed actions as categorical random variables and let the j th element of $\boldsymbol{\theta}_i$ represent the probability that the expert chooses action j at state i . Consequently, $\boldsymbol{\theta}_i$ lies in the $(|\mathcal{A}| - 1)$ -simplex, which we denote by the symbol Δ for brevity of notation, i.e., $\boldsymbol{\theta}_i \in \Delta \subseteq \mathbb{R}^{|\mathcal{A}|}$. The setting is almost identical to the maximum likelihood setting in Section 3.1, with the difference that the deterministic action assignments of the states are replaced with arbitrary distributions over actions. Summarizing all local control parameters in a single matrix, i.e., $\boldsymbol{\Theta} \in \Omega \subseteq \Delta^{|\mathcal{S}|}$, we obtain the global control parameter of the system, which compactly captures the expert behavior (recall Equation 4.1). Note that we denote the global control parameter here by $\boldsymbol{\Theta}$ instead of $\boldsymbol{\omega}$, for reasons that will become clear in Section 4.2.

According to these definitions, each expert action a is characterized by the local policy that is induced by the control parameter of the corresponding state, i.e.

$$\pi(a \mid s = i, \Theta) = \text{CAT}(a \mid \theta_i). \quad (4.2)$$

For simplicity, we write $\pi(a \mid \theta_i)$ from here onwards since the state information is used only to indicate the appropriate local controller.

Considering a finite set of actions, it is convenient to place a (symmetric) Dirichlet prior on the local control parameters, which forms a conjugate distribution to the categorical distribution over actions, i.e.

$$p_\theta(\theta_i \mid \alpha) = \text{DIR}(\theta_i \mid \alpha \cdot \mathbf{1}^{|\mathcal{A}|}).$$

Herein, $\mathbf{1}^{|\mathcal{A}|}$ denotes the vector of all ones of length $|\mathcal{A}|$. The prior is itself parametrized by a concentration parameter α , which can be further described by a hyperprior $p_\alpha(\alpha)$, giving rise to a Bayesian hierarchical model. For simplicity, we assume that the value of α is fixed in this thesis, but the extension to a complete probabilistic treatment is straightforward.

The joint distribution of all model variables is thus given by

$$p(\mathbf{s}, \mathbf{a}, \Theta \mid \alpha) = p_1(s_1) \prod_{i=1}^{|\mathcal{S}|} p_\theta(\theta_i \mid \alpha) \prod_{t=1}^{T-1} \mathcal{T}(s_{t+1} \mid s_t, a_t) \pi(a_t \mid \theta_{s_t}), \quad (4.3)$$

where $\mathbf{a} = (a_1, a_2, \dots, a_{T-1})$ denotes the latent action sequence taken by the expert. The corresponding graphical visualization is depicted in Figure 4.1. For the remainder of this work, we refer to the above as the *static model*.

4.1.1 Gibbs Sampling

Following a Bayesian methodology, our goal is to determine the posterior distribution $p(\Theta \mid \mathbf{s}, \alpha)$, which contains all information necessary to make predictions about the expert behavior. For the static model described in Section 4.1, the required marginalization of the latent action sequence \mathbf{a} can be computed efficiently because the corresponding joint distribution factorizes over time instants. However, for the extended models presented in later sections, a direct marginalization becomes computationally intractable due to the exponential growth of latent variable configurations. For this reason, we follow a sampling-based inference strategy, which is later on generalized to more complex settings.

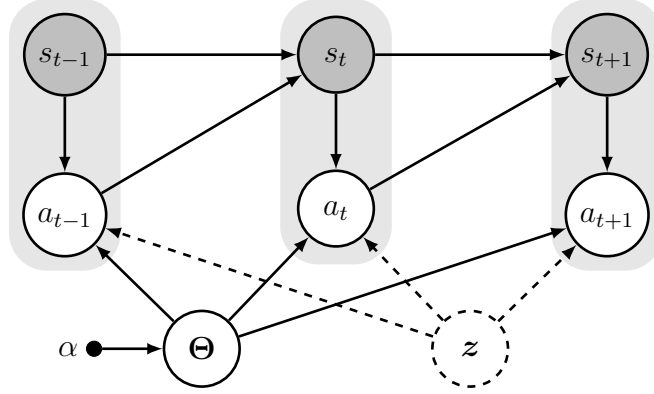


Figure 4.1: Bayesian network representing the policy recognition model. The underlying structure is that of an MDP whose global control parameter Θ is treated as a random variable with prior distribution parametrized by α . The indicator node z is used for the clustering model in Section 4.2. Observed variables are highlighted in gray.

For the simple model described in Equation (4.3), we first approximate the joint posterior distribution $p(\Theta, \mathbf{a} | \mathbf{s}, \alpha)$ over both controllers and actions using a finite number of Q samples and marginalize over \mathbf{a} in a second step, i.e.

$$\begin{aligned} p(\Theta | \mathbf{s}, \alpha) &= \sum_{\mathbf{a}} p(\Theta, \mathbf{a} | \mathbf{s}, \alpha) \\ &\approx \sum_{\mathbf{a}} \left(\frac{1}{Q} \sum_{q=1}^Q \delta_{\Theta^{\{q\}} \mathbf{a}^{\{q\}}}(\Theta, \mathbf{a}) \right) = \frac{1}{Q} \sum_{q=1}^Q \delta_{\Theta^{\{q\}}}(\Theta), \end{aligned} \quad (4.4)$$

where $(\Theta^{\{q\}}, \mathbf{a}^{\{q\}}) \sim p(\Theta, \mathbf{a} | \mathbf{s}, \alpha)$, and $\delta_x(\cdot)$ denotes Dirac's delta function centered at x . This two-step approach gives rise to a simple inference procedure since the joint samples $\{(\Theta^{\{q\}}, \mathbf{a}^{\{q\}})\}_{q=1}^Q$ can be easily obtained through a Gibbs sampling scheme, i.e., by sampling iteratively from the following two conditional distributions,

$$\begin{aligned} p(a_t | \mathbf{a}_{\setminus t}, \mathbf{s}, \Theta, \alpha) &\propto \mathcal{T}(s_{t+1} | s_t, a_t) \pi(a_t | \boldsymbol{\theta}_{s_t}), \\ p(\boldsymbol{\theta}_i | \Theta_{\setminus i}, \mathbf{s}, \mathbf{a}, \alpha) &\propto p_{\theta}(\boldsymbol{\theta}_i | \alpha) \prod_{t: s_t=i} \pi(a_t | \boldsymbol{\theta}_i). \end{aligned} \quad (4.5)$$

Herein, $\mathbf{a}_{\setminus t}$ and $\Theta_{\setminus i}$ refer to all actions/controllers except a_t and $\boldsymbol{\theta}_i$, respectively. The latter expression reveals that, in order to sample $\boldsymbol{\theta}_i$, we need to consider only those actions played at the corresponding state i , i.e., $\{a_t : s_t = i\}$. Furthermore, the first expression shows that, given Θ , all actions can be sampled independently of each other. Therefore, inference can be done in parallel for all $\{\boldsymbol{\theta}_i\}$. This can be also seen from the

posterior distribution of the global control parameter Θ , which factorizes over states, i.e.

$$p(\Theta | \mathbf{s}, \mathbf{a}, \alpha) \propto \prod_{i=1}^{|\mathcal{S}|} p_{\theta}(\theta_i | \alpha) \prod_{t: s_t=i} \pi(a_t | \theta_i). \quad (4.6)$$

From the conjugacy of $p_{\theta}(\theta_i | \alpha)$ and $\pi(a_t | \theta_i)$, it follows that the posterior over θ_i is again Dirichlet distributed with updated concentration parameter. In particular, denoting by $\phi_{i,j}$ the number of times that action j is played at state i in the current assignment of actions \mathbf{a} , i.e.

$$\phi_{i,j} \triangleq \sum_{t: s_t=i} \mathbb{1}(a_t = j), \quad (4.7)$$

and by collecting these quantities in the form of vectors, i.e., $\phi_i \triangleq [\phi_{i,1}, \dots, \phi_{i,|\mathcal{A}|}]$, we can rewrite Equation (4.6) as

$$p(\Theta | \mathbf{s}, \mathbf{a}, \alpha) = \prod_{i=1}^{|\mathcal{S}|} \text{DIR}(\theta_i | \phi_i + \alpha \cdot \mathbf{1}^{|\mathcal{A}|}). \quad (4.8)$$

4.1.2 Collapsed Gibbs Sampling

Choosing a Dirichlet distribution as prior model for the local controllers $\{\theta_i\}$ is convenient as it allows us to arrive at closed-form expressions for the conditional distributions in Equation (4.5), which is required to efficiently run the Gibbs sampler. As an alternative to the described two-step approach, we can also exploit the conjugacy property of $p_{\theta}(\theta_i | \alpha)$ and $\pi(a_t | \theta_i)$ to marginalize out the control parameters *during* the sampling process, giving rise to a collapsed sampling scheme.

Collapsed sampling is advantageous in two ways: first, it reduces the total number of variables to be sampled and, hence, the number of computations required per Gibbs iteration; second, it increases the mixing speed of the underlying Markov chain that governs the sampling process, reducing the correlation of the obtained samples and, with it, the variance of the resulting policy estimate.

Formally, collapsing means that we replace the approximation of the joint distribution $p(\Theta, \mathbf{a} | \mathbf{s}, \alpha)$ in Equation (4.4) with an approximation of the *marginal* distribu-

tion $p(\mathbf{a} \mid \mathbf{s}, \alpha)$, i.e.

$$\begin{aligned} p(\boldsymbol{\Theta} \mid \mathbf{s}, \alpha) &= \sum_{\mathbf{a}} p(\boldsymbol{\Theta} \mid \mathbf{s}, \mathbf{a}, \alpha) p(\mathbf{a} \mid \mathbf{s}, \alpha) \\ &\approx \sum_{\mathbf{a}} p(\boldsymbol{\Theta} \mid \mathbf{s}, \mathbf{a}, \alpha) \left(\frac{1}{Q} \sum_{q=1}^Q \delta_{\mathbf{a}^{\{q\}}}(\mathbf{a}) \right) = \frac{1}{Q} \sum_{q=1}^Q p(\boldsymbol{\Theta} \mid \mathbf{s}, \mathbf{a}^{\{q\}}, \alpha), \end{aligned} \quad (4.9)$$

where $\mathbf{a}^{\{q\}} \sim p(\mathbf{a} \mid \mathbf{s}, \alpha)$. In contrast to the previous approach, the target distribution $p(\boldsymbol{\Theta} \mid \mathbf{s}, \alpha)$ is no longer represented by a sum of Dirac measures (Equation 4.4) but described by a product of Dirichlet mixtures (compare Equation 4.8). The required samples $\{\mathbf{a}^{\{q\}}\}$ can be obtained from a collapsed Gibbs sampler according to

$$\begin{aligned} p(a_t \mid \mathbf{a}_{\setminus t}, \mathbf{s}, \alpha) &\propto \int_{\Delta^{|\mathbf{s}|}} p(\mathbf{s}, \mathbf{a}, \boldsymbol{\Theta} \mid \alpha) d\boldsymbol{\Theta} \\ &\propto \mathcal{T}(s_{t+1} \mid s_t, a_t) \int_{\Delta} p_{\theta}(\boldsymbol{\theta}_{s_t} \mid \alpha) \prod_{t': s_{t'} = s_t} \pi(a_{t'} \mid \boldsymbol{\theta}_{s_t}) d\boldsymbol{\theta}_{s_t}. \end{aligned}$$

It turns out that the above distribution provides an easy sampling mechanism since the integral part, when viewed as a function of action a_t only, can be identified as the conditional of a Dirichlet-multinomial distribution, which is then reweighted by the likelihood $\mathcal{T}(s_{t+1} \mid s_t, a_t)$ of the observed state transition. The final (unnormalized) weights of the resulting categorical distribution are hence given by

$$p(a_t = j \mid \mathbf{a}_{\setminus t}, \mathbf{s}, \alpha) \propto \mathcal{T}(s_{t+1} \mid s_t, a_t = j) \cdot (\varphi_{t,j} + \alpha), \quad (4.10)$$

where $\varphi_{t,j}$ counts the number of occurrences of action j among all actions in $\mathbf{a}_{\setminus t}$ played at the same state as a_t (that is, s_t). Explicitly,

$$\varphi_{t,j} \triangleq \sum_{\substack{t': s_{t'} = s_t \\ t' \neq t}} \mathbb{1}(a_{t'} = j).$$

Note that these values can be also expressed in terms of the sufficient statistics for the ordinary Gibbs sampler introduced in Equation (4.7), i.e.

$$\varphi_{t,j} = \phi_{s_t,j} - \mathbb{1}(a_t = j).$$

As before, actions played at different states can be sampled independently of each other because they are generated by different local controllers. Consequently, inference about $\boldsymbol{\Theta}$ again decouples for all system states.

4.2 Toward Large State Spaces: A Clustering Approach

While the methodology introduced so far allows to solve the policy recognition problem in finite domains, the presented approaches quickly become infeasible for larger problems as the number of parameters to be learned (i.e., the size of Θ) grows unbounded with the size of the state space. As a consequence, the presented model is prone to overfitting because, for large enough \mathcal{S} , no demonstration set will be rich enough to represent all situations that can occur to the agent during the decision-making process sufficiently well. A particular problem—which we already encountered with the ML approach in Chapter 3—is that the static model makes no assumptions on the structure of Θ but treats all local policies separately. Hence, the model is unable to generalize the expert behavior to regions of the state space where no demonstrations are available.

A simple way to counteract both problems and scale the approach to larger domains is to restrict the complexity of the targeted behavior class by providing only a finite number of policy parameters that need to be shared across states. In the context of the decision-making process, this corresponds to the assumption that, at each state, the expert selects an action according to one of K local policies, described by the parameters $\{\theta_k\}_{k=1}^K$. To model this situation, we introduce a set of *indicator* or *cluster assignment variables*, i.e., $\{z_i\}_{i=1}^{|\mathcal{S}|}$, $z_i \in \{1, \dots, K\}$, which map the states to the corresponding local controllers. The resulting assignment induces a partitioning of the state space (Figure 4.2), giving rise to the following K state clusters,

$$\mathcal{C}_k \triangleq \{i : z_i = k\}, \quad k \in \{1, \dots, K\}.$$

The joint distribution in Equation (4.3) changes accordingly to

$$p(\mathbf{s}, \mathbf{a}, \mathbf{z}, \Theta | \alpha) = p_1(s_1) \prod_{k=1}^K p_{\theta}(\theta_k | \alpha) \prod_{t=1}^{T-1} \mathcal{T}(s_{t+1} | s_t, a_t) \pi(a_t | \theta_{z_{s_t}}) p_z(\mathbf{z}), \quad (4.11)$$

where $\mathbf{z} = (z_1, z_2, \dots, z_{|\mathcal{S}|})$ denotes the collection of all indicator variables (represented by the dashed node in Figure 4.1), and $p_z(\mathbf{z})$ is the corresponding prior distribution to be further discussed in Section 4.2.3. Note that the static model in Equation (4.3) can be recovered as a special case of the above when each state describes its own cluster, i.e., by setting $K = |\mathcal{S}|$ and fixing $z_i \triangleq i$ (hence the name *static*).

In contrast to the static approach, both the indicator z_i and the corresponding control parameter θ_{z_i} are required in order to characterize the expert's behavior at a given state i . Accordingly, the global control parameter of the model is given by $\omega \triangleq (\Theta, \mathbf{z})$

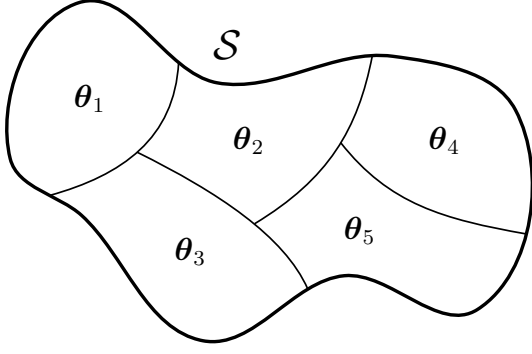


Figure 4.2: Schematic illustration of the clustering model, which uses a finite number of controllers to explain the expert behavior. The state space \mathcal{S} is partitioned into a set of clusters $\{\mathcal{C}_k\}$, each governed by its own local control parameter θ_k .

with underlying parameter space $\Omega \subseteq \Delta^K \times \{1, \dots, K\}^{|\mathcal{S}|}$, and our target distribution becomes $p(\Theta, \mathbf{z} | \mathbf{s}, \alpha)$. In what follows, we derive the Gibbs and the collapsed Gibbs sampler as mechanisms for approximate inference in this setting.

4.2.1 Gibbs Sampling

The expressions for the conditional distributions over actions and local controllers take a similar form to those of the static model (see Section 4.1.1), as shown by the following equations. Here, the only difference is that actions are no longer grouped by states but according to their generating local policies or, equivalently, the clusters $\{\mathcal{C}_k\}$, i.e.

$$p(a_t | \mathbf{a}_{\setminus t}, \mathbf{s}, \mathbf{z}, \Theta, \alpha) \propto \mathcal{T}(s_{t+1} | s_t, a_t) \cdot \pi(a_t | \theta_{z_{s_t}}),$$

$$p(\theta_k | \Theta_{\setminus k}, \mathbf{s}, \mathbf{a}, \mathbf{z}, \alpha) \propto p_\theta(\theta_k | \alpha) \prod_{t: z_{s_t}=k} \pi(a_t | \theta_k) = p_\theta(\theta_k | \alpha) \prod_{t: s_t \in \mathcal{C}_k} \pi(a_t | \theta_k).$$

The latter expression, again, takes the form of a Dirichlet distribution with updated concentration parameter, i.e.

$$p(\theta_k | \Theta_{\setminus k}, \mathbf{s}, \mathbf{a}, \mathbf{z}, \alpha) = \text{DIR}(\theta_k | \boldsymbol{\xi}_k + \alpha \cdot \mathbf{1}^{|\mathcal{A}|}),$$

with $\boldsymbol{\xi}_k \triangleq [\xi_{k,1}, \dots, \xi_{k,|\mathcal{A}|}]$, where $\xi_{k,j}$ denotes the number of times that action j is played at states belonging to cluster \mathcal{C}_k in the current assignment of \mathbf{a} . Explicitly,

$$\xi_{k,j} \triangleq \sum_{t: z_{s_t}=k} \mathbf{1}(a_t = j) = \sum_{i \in \mathcal{C}_k} \sum_{t: s_t=i} \mathbf{1}(a_t = j), \quad (4.12)$$

which is nothing but the sum of the $\phi_{i,j}$'s (Equation 4.7) of the corresponding states, i.e.

$$\xi_{k,j} = \sum_{i \in \mathcal{C}_k} \phi_{i,j}. \quad (4.13)$$

In addition to the actions and control parameters, we also need to sample the new indicator variables $\{z_i\}_{i=1}^{|S|}$, whose conditional distributions can be expressed in terms of the conditional prior model $p(z_i | \mathbf{z}_{\setminus i})$ and the likelihood of the triggered actions, i.e.

$$p(z_i | \mathbf{z}_{\setminus i}, \mathbf{s}, \mathbf{a}, \Theta, \alpha) \propto p(z_i | \mathbf{z}_{\setminus i}) \prod_{t: s_t=i} \pi(a_t | \theta_{z_i}). \quad (4.14)$$

4.2.2 Collapsed Gibbs Sampling

Analogous to Section 4.1.2, we derive the collapsed Gibbs sampler by marginalizing out the local control parameters $\{\theta_k\}$, i.e.

$$\begin{aligned} p(z_i | \mathbf{z}_{\setminus i}, \mathbf{s}, \mathbf{a}, \alpha) &\propto \int_{\Delta^K} p(\mathbf{s}, \mathbf{a}, \mathbf{z}, \Theta | \alpha) d\Theta \\ &\propto p(z_i | \mathbf{z}_{\setminus i}) \int_{\Delta^K} \prod_{k=1}^K p_\theta(\theta_k | \alpha) \prod_{t=1}^{T-1} \pi(a_t | \theta_{z_{s_t}}) d\Theta \\ &\propto p(z_i | \mathbf{z}_{\setminus i}) \int_{\Delta^K} \prod_{k=1}^K p_\theta(\theta_k | \alpha) \prod_{i'=1}^{|S|} \prod_{t: s_t=i'} \pi(a_t | \theta_{z_{i'}}) d\Theta \\ &\propto p(z_i | \mathbf{z}_{\setminus i}) \int_{\Delta^K} \prod_{k=1}^K p_\theta(\theta_k | \alpha) \prod_{i': z_{i'}=k} \prod_{t: s_t=i'} \pi(a_t | \theta_k) d\Theta \\ &\propto p(z_i | \mathbf{z}_{\setminus i}) \prod_{k=1}^K \left(\int_{\Delta} p_\theta(\theta_k | \alpha) \prod_{t: s_t \in \mathcal{C}_k} \pi(a_t | \theta_k) d\theta_k \right). \end{aligned} \quad (4.15)$$

Here, we first grouped the actions by their associated states and then grouped the states themselves by the clusters $\{\mathcal{C}_k\}$.

Again, the conditional distribution admits an easy sampling mechanism as it takes the form of a product of Dirichlet-multinomials, reweighted by the conditional prior distribution over indicators $p(z_i | \mathbf{z}_{\setminus i})$. In particular, we observe that all actions played at some state i appear in exactly one of the K integrals of the last equation. In other words, by changing the value of z_i (i.e., by assigning state i to another cluster), only two of the involved integrals are affected: the one belonging to the previously assigned cluster, and the one of the new cluster. Inference about the value of z_i can thus be carried out using the following two sets of sufficient statistics:

- $\phi_{i,j}$: the number of times action j is played at state i ,
- $\psi_{i,j,k}$: the number of times action j is played at states assigned to cluster \mathcal{C}_k , excluding state i .

The $\phi_{i,j}$'s are the same as in Equation (4.7) and their definition is repeated here just as a reminder. For the $\psi_{i,j,k}$'s, on the other hand, we find the following explicit expression,

$$\psi_{i,j,k} \triangleq \sum_{\substack{i' \in \mathcal{C}_k \\ i' \neq i}} \sum_{t: s_t = i'} \mathbb{1}(a_t = j),$$

which can be also written in terms of the statistics $\xi_{k,j}$ for the ordinary Gibbs sampler defined in Equation (4.12), i.e.

$$\psi_{i,j,k} = \xi_{k,j} - \mathbb{1}(i \in \mathcal{C}_k) \cdot \phi_{i,j}.$$

Collecting these quantities in a vector, i.e., $\boldsymbol{\psi}_{i,k} \triangleq [\psi_{i,1,k}, \dots, \psi_{i,|\mathcal{A}|,k}]$, we end up with the following simplified expression,

$$p(z_i = k \mid \mathbf{z}_{\setminus i}, \mathbf{s}, \mathbf{a}, \alpha) \propto p(z_i = k \mid \mathbf{z}_{\setminus i}) \prod_{k'=1}^K \text{DIRMULT}(\boldsymbol{\psi}_{i,k'} + \mathbb{1}(k' = k) \cdot \boldsymbol{\phi}_i \mid \alpha). \quad (4.16)$$

Further, we obtain the following result for the conditional distribution of action a_t ,

$$p(a_t \mid \mathbf{a}_{\setminus t}, \mathbf{s}, \mathbf{z}, \alpha) \propto \mathcal{T}(s_{t+1} \mid s_t, a_t) \int_{\Delta} p_{\theta}(\boldsymbol{\theta}_{z_{s_t}} \mid \alpha) \prod_{t': z_{s_{t'}} = z_{s_t}} \pi(a_{t'} \mid \boldsymbol{\theta}_{z_{s_t}}) d\boldsymbol{\theta}_{z_{s_t}}.$$

By introducing the sufficient statistics $\{\vartheta_{t,j}\}$, which count the number of occurrences of action j among all states that are assigned to the same cluster as s_t (i.e., the cluster $\mathcal{C}_{z_{s_t}}$), excluding a_t itself, i.e.

$$\vartheta_{t,j} \triangleq \sum_{\substack{t': z_{s_{t'}} = z_{s_t} \\ t' \neq t}} \mathbb{1}(a_{t'} = j),$$

we arrive at the final expression

$$p(a_t = j \mid \mathbf{a}_{\setminus j}, \mathbf{s}, \mathbf{z}, \alpha) \propto (\vartheta_{t,j} + \alpha) \cdot \mathcal{T}(s_{t+1} \mid s_t, a_t = j).$$

As for the static model, we can further establish a relationship between the statistics used for the ordinary and the collapsed sampler, i.e.

$$\vartheta_{t,j} = \xi_{z_{s_t},j} - \mathbb{1}(a_t = j).$$

4.2.3 Prior Models

In order to complete our model, we need to specify a prior distribution over indicator variables $p_z(\mathbf{z})$. The following paragraphs present three candidate models:

Non-Informative Prior

The simplest prior model is the non-informative prior over partitionings. It reflects the assumption that, a priori, all cluster assignments are equally likely and that the indicators $\{z_i\}$ are mutually independent. In this case, $p_z(\mathbf{z})$ is constant and the term $p(z_i | \mathbf{z}_{\setminus i})$ in Equations (4.14) and (4.15) disappears, so that the conditional distribution of indicator z_i becomes directly proportional to the likelihood of the inferred action sequence.

Mixing Prior

Another simple yet more expressive prior model can be realized by using a (finite) Dirichlet mixture. Instead of assuming that the indicator variables are independent, the model describes their relationship through set of mixing coefficients $\mathbf{q} \triangleq [q_1, \dots, q_K]$, where $q_k \in [0, 1]$ represents the prior probability that an indicator variable takes on value k . The mixing coefficients are themselves modeled by a Dirichlet distribution, which finally yields

$$\begin{aligned} \mathbf{q} &\sim \text{DIR}(\mathbf{q} | \frac{\gamma}{K} \cdot \mathbf{1}^K), \\ z_i | \mathbf{q} &\sim \text{CAT}(z_i | \mathbf{q}), \end{aligned} \tag{4.17}$$

where $\gamma \in (0, \infty)$ is a concentration parameter that controls the variability of the mixing coefficients.

Note that the indicator variables $\{z_i\}$ are still *conditionally* independent given the mixing coefficients in this model. More specifically, for a fixed \mathbf{q} , the conditional distribution of a single indicator in Equations (4.14) and (4.15) takes the following simple form,

$$p(z_i = k | \mathbf{z}_{\setminus i}, \mathbf{q}) = q_k.$$

Since the value of \mathbf{q} is typically unknown, we have two options to include the model into our framework. The first option is to sample \mathbf{q} in addition to the remaining variables by drawing values from the following conditional distribution during the Gibbs procedure,

$$\begin{aligned} p(\mathbf{q} | \mathbf{s}, \mathbf{a}, \mathbf{z}, \boldsymbol{\Theta}, \alpha) &\propto \text{DIR}(\mathbf{q} | \frac{\gamma}{K} \cdot \mathbf{1}^K) \prod_{i=1}^{|\mathcal{S}|} \text{CAT}(z_i | \mathbf{q}) \\ &\propto \text{DIR}(\mathbf{q} | \boldsymbol{\zeta} + \frac{\gamma}{K} \cdot \mathbf{1}^K), \end{aligned}$$

with $\boldsymbol{\zeta} \triangleq [\zeta_1, \dots, \zeta_K]$, where ζ_k denotes the number of assignments that map to cluster \mathcal{C}_k , i.e.

$$\zeta_k \triangleq \sum_{i=1}^{|\mathcal{S}|} \mathbb{1}(z_i = k).$$

Alternatively, we can marginalize out the mixing proportions \mathbf{q} during the inference process, as we did with the control parameters in Sections 4.1.2 and 4.2.2. The result is (additional) collapsing in \mathbf{q} . To this end, we replace the factor $p(z_i = k | \mathbf{z}_{\setminus i})$ appearing in Equations (4.15) and (4.16) with

$$p(z_i = k | \mathbf{z}_{\setminus i}, \gamma) \propto (\zeta_k^{(\setminus i)} + \frac{\gamma}{K}), \quad (4.18)$$

where $\zeta_k^{(\setminus i)}$ is defined like ζ_k but without counting the current value of indicator z_i , i.e.

$$\zeta_k^{(\setminus i)} \triangleq \sum_{\substack{i'=1 \\ i' \neq i}}^{|\mathcal{S}|} \mathbb{1}(z_{i'} = k) = \zeta_k - \mathbb{1}(z_i = k).$$

A detailed derivation is omitted here but follows the same line of argument as for the collapsing of $\boldsymbol{\Theta}$ in Section 4.1.2.

Spatial Prior

Both previous prior models assume (conditional) independence of the indicator variables, and hence, they make no specific assumptions about the spatial relationships of the agent’s local policies at different states. As an alternative, we now consider a prior model that explicitly establishes a spatial dependence between the policy assignments, thereby promoting a particular type of state clustering. A reasonable choice is to use a model that preferably groups “similar” states together and assigns those states the same local control parameter. As demonstrated by the MAP approach in Chapter 3, this will help us to extrapolate the expert behavior to regions where no demonstration is available.

In the following, we express the similarity of states with the help of a monotonically decreasing decay function $f : [0, \infty) \rightarrow [0, 1]$, which takes as input the distance between the two states and returns a positive real-valued similarity score. The state distances are assumed to be given through a distance metric $\chi : \mathcal{S} \times \mathcal{S} \rightarrow [0, \infty)$. Details on how such a metric can be defined in an arbitrary scenario will be provided in Section 10.2.1, where we establish a relationship between system states based on the underlying system dynamics; for now, we simply assume that a suitable χ is given.

Reusing our ideas from Section 3.2, we construct the prior in the form of a Potts model, which induces a spatial dependence between the local policies of the expert, i.e.

$$p_z(\mathbf{z}) \propto \prod_{i=1}^{|\mathcal{S}|} \exp \left(\frac{\beta}{2} \sum_{\substack{j=1 \\ j \neq i}}^{|\mathcal{S}|} f(\Delta_{i,j}) \delta(z_i, z_j) \right). \quad (4.19)$$

Here, the simple averaging scheme from Equation (3.5) has been replaced with a weighted average that explicitly takes into account the pairwise state distances $\Delta_{i,j} \triangleq \chi(s_i, s_j)$, $i, j \in \{1, \dots, |\mathcal{S}|\}$. The conditional distribution of a single indicator variable z_i is then obtained as

$$p(z_i | \mathbf{z}_{\setminus i}) \propto \exp \left(\beta \sum_{\substack{j=1 \\ j \neq i}}^{|\mathcal{S}|} f(\Delta_{i,j}) \delta(z_i, z_j) \right), \quad (4.20)$$

where the scaling factor $\frac{1}{2}$ has again canceled, due to the symmetry property of χ (compare Equation 3.7). This completes our inference framework for finite spaces.

4.3 Countably Infinite and Uncountable State Spaces

The main advantage of the clustering approach presented in Section 4.2 is that, due to the limited number of local policies to be learned from the finite amount of demonstration data, the existing inference methodology can be applied to state spaces of arbitrary size—including countably infinite and uncountable state spaces. This extension had been practically impossible for the static model because of the overfitting problem explained at the beginning of in Section 4.2.

Nevertheless, there remains a fundamental conceptual challenge: a direct application of the model to infinite spaces would imply that the distribution over possible state partitionings becomes an infinite-dimensional object (i.e., in the case of uncountable state spaces, a distribution over functional mappings from states to local controllers), requiring an infinite number of indicator variables. While there exist possibilities to model the statistical relationship between such infinite collections of variables (e.g., using thinned completely random measures [Fot+13; FW15], see *Discussion and Outlook*), a detailed treatment of these models goes beyond the scope of this thesis and will be left for future work.

Instead, we follow a simpler strategy that builds upon the concepts we have already established so far. The approach is based on the following trivial fact: even if the number of latent cluster assignments grows unbounded for large system sizes, the

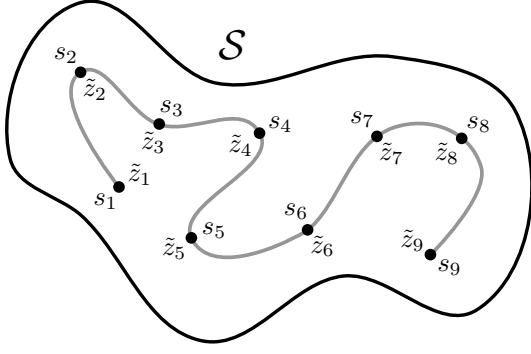


Figure 4.3: Illustration of the reduced state space model, which operates on the space $\tilde{\mathcal{S}} = \{s_1, s_2, \dots, s_T\}$ of visited trajectory states. Note that the underlying decision-making process is assumed to be discrete in time; the continuous gray line shown in the figure is only to highlight the temporal ordering of the trajectory states.

amount of observed trajectory data always remains *finite*. A simple solution to the modeling problem is, therefore, to reformulate the inference task on a reduced state space $\tilde{\mathcal{S}} \triangleq \{s_1, s_2, \dots, s_T\}$, which contains only the states along the observed expert trajectories (Figure 4.3).

Reducing the state space in this way means that we need to consider only a finite set of indicator variables, one for each expert state in $\tilde{\mathcal{S}}$, which always induces a model of finite size. To distinguish from the previous indicator set, we denote these new variables by \tilde{z} . Assuming that no state is visited twice by the expert, we may use the same index set $\{1, \dots, T\}$ for both, indicators and states, i.e., $\tilde{z} \triangleq \{\tilde{z}_1, \dots, \tilde{z}_T\}$.*

In order to limit the complexity of the corresponding prior model for larger data sets, we let the value of indicator \tilde{z}_t depend only on a subset of the remaining variables $\tilde{z}_{\setminus t}$ as defined by some neighborhood rule \mathcal{N}_t . The resulting joint distribution of assignments is then given by

$$p_{\tilde{z}}(\tilde{z} | \mathbf{s}) \propto \prod_{t=1}^T \exp \left(\frac{\beta}{2} \sum_{t' \in \mathcal{N}_t} f(\Delta_{t,t'}) \delta(\tilde{z}_t, \tilde{z}_{t'}) \right),$$

which now implicitly depends on the state sequence \mathbf{s} through the pairwise distances $\Delta_{t,t'} \triangleq \chi(s_t, s_{t'})$, $t, t' \in \{1, \dots, T\}$ —hence the conditioning on \mathbf{s} .

4.3.1 Marginal Invariance

The use of a finite number of indicator variables along the observed expert trajectories obviously circumvents the above-mentioned problem of representational complexity.

* Note that we make this assumption for notational convenience only and that it is not required from a mathematical point of view. Nonetheless, for uncountable state spaces the assumption is reasonable since the event of reaching the exact same state twice has measure zero for most dynamic models. In the general case, however, the indicator variables require their own index set to ensure that each system state is associated with exactly one cluster, even when visited multiple times. This situation changes, when the expert behavior changes over time (see Section 10.3).

Nevertheless, there are some caveats associated with this approach. First of all, using a reduced state space model raises the question of marginal invariance [BF11; FW15]: if we added a new trajectory point to the data set, would it change our belief about the expert policy at previous states? In particular, how is this situation different from modeling the new data point together with the initial trajectory data in the first place? And furthermore, what does such a reduced model imply for unvisited states? Can we still use it to make predictions about their local policies? These questions are, in fact, important for generalizing the expert demonstrations to new situations. However, for modeling the expert behavior based on a fixed data set, these questions are less relevant; hence, the detailed discussion of the issue is deferred to Appendix A.

4.3.2 A Factor Graph Model

A second albeit related issue caused by the reduced modeling approach is that we lose the simple causal interpretation of the data generation process described in Section 2.1.2. In the finite state space case, we could think of a trajectory as being constructed by the following step-wise mechanism: first, the prior $p_z(\mathbf{z})$ is used to generate a set of indicator variables for all states. Independently, we pick some value for α from $p_\alpha(\alpha)$ and sample K local control parameters from $p_\theta(\boldsymbol{\theta}_k | \alpha)$. To generate a trajectory, we start with an initial state s_1 , generated by $p_1(s_1)$, select a random action a_1 from $\pi(a_1 | s_1, \boldsymbol{\theta}_{z_{s_1}})$ and transition to a new state s_2 according to $\mathcal{T}(s_2 | s_1, a_1)$, where we select another action a_2 , and so on. Such a directed way of thinking is possible since the finite model naturally obeys a causal structure where later states depend on earlier ones and the decisions made there. Furthermore, the cluster assignments $\{z_i\}$ and local controllers $\{\boldsymbol{\theta}_k\}$ could be generated in advance and independently of each other because they were assumed marginally independent by the model.

For the reduced state space model, this interpretation no longer applies as it has no natural directionality. In fact, its variables depend on each other in a cyclic fashion: altering the value of a particular indicator variable (say, the one corresponding to the last trajectory point) will have an effect on the values of all remaining indicators due to their spatial relationship encoded by the “prior distribution” $p_z(\mathbf{z} | \mathbf{s})$. Changing the values of the other indicators, however, will influence the actions being played at the respective states which, in turn, alters the probability of ending up with the observed trajectory in the first place and, hence, the position and value of the indicator variable we started with. Explaining the data generation of this model using a simple generative process is, therefore, not possible.

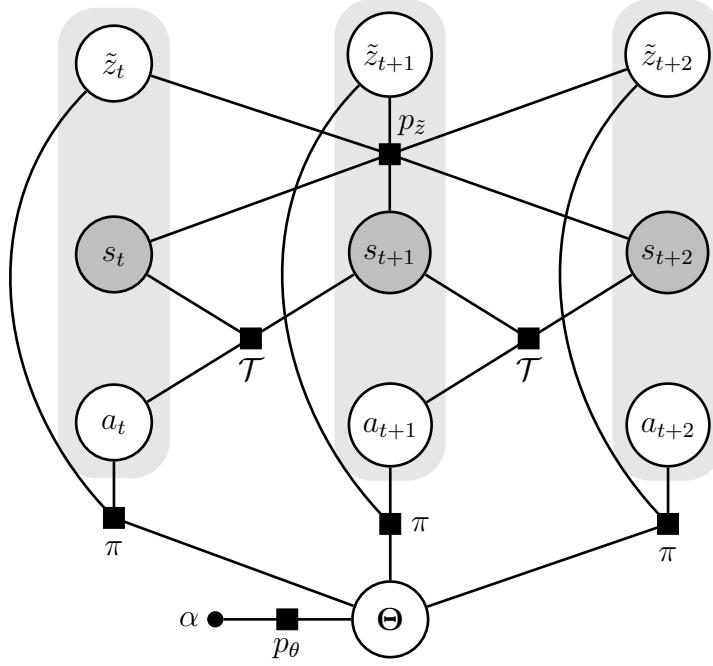


Figure 4.4: Factor graph of the reduced state space model shown in Figure 4.3, illustrating the circular dependence of the involved variables. The factors are defined by the same building blocks that are used for the finite state space model in Equation (4.11). Shaded nodes correspond to observed variables.

Nevertheless, the individual building blocks of the model (that is, the policy, the transition model, etc.) together form a valid distribution over the model variables, which can be readily used for parameter inference. For the reasons explained above, it makes sense to define this distribution in the form of a discriminative model, ignoring the underlying generative aspects of the process. This is sufficient since we can always condition on the observed state sequence \mathbf{s} , i.e.

$$p(\mathbf{a}, \Theta, \mathbf{z} | \mathbf{s}, \alpha) = \frac{1}{Z_{\mathbf{s}}} p_z(\mathbf{z} | \mathbf{s}) \prod_{k=1}^K p_{\theta}(\theta_k | \alpha) p_1(s_1) \prod_{t=1}^{T-1} \mathcal{T}(s_{t+1} | s_t, a_t) \pi(a_t | \theta_{z_{s_t}}). \quad (4.21)$$

Herein, $Z_{\mathbf{s}}$ is the corresponding data-dependent normalizing constant. The structure of this distribution is illustrated by the factor graph shown in Figure 4.4, which highlights the circular dependence of the variables.

Note that, for any *fixed* state sequence \mathbf{s} , this distribution indeed encodes the same basic properties as the finite model in Equation (4.11). In particular, the conditional distributions of all remaining variables remain unchanged, which allows us to apply the same inference machinery that we already used in the finite case. For a deeper discussion on the difference between the two models, we again point to Appendix A.

5

Nonparametric Policy Recognition

In the Chapter 4, we presented a probabilistic policy recognition framework for modeling the expert behavior using a finite mixture of K local policies. Basically, there are two situations when such a model is useful:

- either, we know the true number of expert policies,
- or, irrespective of the true behavioral complexity, we want to find an approximate system description in terms of at most K distinct control situations (compare finite state controllers [Meu+99]).

In all other cases, we are faced with the nontrivial problem of choosing K .

By selecting a certain value for K , we can directly control the hypothesis class of considered expert behaviors. However, the choice of K should not only be regarded a mathematical necessity to perform inference in our model. From a system identification point of view, it seems indeed more reasonable to infer the required granularity of the state partitioning from the observed expert behavior itself, instead of enforcing a particular model order. This way, we can gain valuable information about the underlying control structure and state representation used by the expert, which offers a possibility to learn a state partitioning of task-appropriate complexity directly from the demonstration data.

From a statistical modeling perspective, there are two common ways to approach this problem. The first is to make use of model order selection techniques, which allows us to determine the most parsimonious model that is in agreement with the observed data. However, focusing on a particular model order means that we consider only one possible explanation of the demonstrated behavior. A more elegant approach is to keep the complexity of the model flexible and, hence, adaptable to the data. Mathematically, this can be achieved by assuming a potentially infinite set of model parameters, of which only a finite subset is required to explain the particular data set at hand. This alternative way of thinking opens the door to the rich class of nonparametric models, which provide an integrated framework to formulate the inference problem over both model parameters and model complexity as a joint learning problem.

5.1 A Dirichlet Process Mixture Model

The classical way to nonparametric clustering is to use a Dirichlet process mixture model (DPMM) [Nea00]. These models can be obtained by starting from a finite Dirichlet mixture and letting the number of mixture components (in our case, the number of local controllers) approach infinity.

In the context of our policy recognition problem, we start with the clustering model from Section 4.2 using a mixing prior (Equation 4.17) over indicator variables, i.e.

$$\begin{aligned}
\mathbf{q} &\sim \text{DIR}(\mathbf{q} \mid \frac{\gamma}{K} \cdot \mathbf{1}^K) & z_i \mid \mathbf{q} &\sim \text{CAT}(z_i \mid \mathbf{q}) \\
\boldsymbol{\theta}_k &\sim \text{DIR}(\boldsymbol{\theta}_k \mid \alpha \cdot \mathbf{1}^{|A|}) & a_t \mid s_t, \boldsymbol{\Theta}, \mathbf{z} &\sim \pi(a_t \mid \boldsymbol{\theta}_{z_{s_t}}) \\
s_1 &\sim p_1(s_1) & s_{t+1} \mid s_t, a_t &\sim \mathcal{T}(s_{t+1} \mid s_t, a_t) .
\end{aligned} \tag{5.1}$$

From these relations, we arrive at the corresponding nonparametric model as K goes to infinity. For the theoretical foundations of this limit, the reader is referred to the more general literature on Dirichlet processes, such as [Fer73; Nea00]. Here, we restrict ourselves to providing the resulting sampling mechanisms to solve the inference problem.

In a DPMM, the mixing proportions \mathbf{q} of the local parameters $\{\boldsymbol{\theta}_k\}$ are marginalized out, giving rise to a collapsed sampling strategy. The resulting distribution over partitionings is described by the Chinese restaurant process (CRP) [Ald85], which can be derived by considering the limit $K \rightarrow \infty$ of the mixing process described by the Gibbs update in Equation (4.18), i.e.

$$p(z_i = k \mid \mathbf{z}_{\setminus i}, \gamma) \propto \begin{cases} \zeta_k^{(\setminus i)} & \text{if } k \in \{1, \dots, K^*\}, \\ \gamma & \text{if } k = K^* + 1. \end{cases} \tag{5.2}$$

Here, K^* denotes the number of distinct entries in $\mathbf{z}_{\setminus i}$, which are represented by the numerical values $\{1, \dots, K^*\}$. In this model, a state joins an existing cluster (i.e., a group of states whose indicators share the same value) with probability proportional to the number of states already contained in that cluster. Alternatively, it creates a new cluster with probability proportional to γ .

From the relations in (5.1) it is evident that, fixing a particular setting of indicators \mathbf{z} , the conditional distributions of all other variable types remain unchanged compared to those of the finite cluster model in Section 4.2 — we only replaced the prior model $p_z(\mathbf{z})$ with the CRP. Hence, we can apply the same Gibbs updates for the actions and

controllers as in Chapter 4 and need to rederive only the conditional distributions of the indicator variables under consideration of the CRP model. According to Equation (5.2), we herein need to distinguish whether an indicator variable takes a value already occupied by other indicators (i.e., it joins an existing cluster) or it is assigned a new value (i.e., it creates a new cluster).

Let $\{\boldsymbol{\theta}_k\}_{k=1}^{K^*}$ denote the set of control parameters associated with $\mathbf{z}_{\setminus i}$. For the first case ($k \in \{1, \dots, K^*\}$), we can then write

$$\begin{aligned}
 p(z_i = k \mid \mathbf{z}_{\setminus i}, \mathbf{s}, \mathbf{a}, \{\boldsymbol{\theta}_{k'}\}_{k'=1}^{K^*}, \alpha, \gamma) \\
 &= p(z_i = k \mid \mathbf{z}_{\setminus i}, \{a_t\}_{t:s_t=i}, \boldsymbol{\theta}_k, \alpha, \gamma) \\
 &\propto p(z_i = k \mid \mathbf{z}_{\setminus i}, \boldsymbol{\theta}_k, \alpha, \gamma) p(\{a_t\}_{t:s_t=i} \mid z_i = k, \mathbf{z}_{\setminus i}, \boldsymbol{\theta}_k, \alpha, \gamma) \\
 &\propto p(z_i = k \mid \mathbf{z}_{\setminus i}, \gamma) p(\{a_t\}_{t:s_t=i} \mid \boldsymbol{\theta}_k) \\
 &\propto \zeta_k^{(\setminus i)} \cdot \prod_{t:s_t=i} \pi(a_t \mid \boldsymbol{\theta}_k).
 \end{aligned}$$

For the second case ($k = K^* + 1$), we instead obtain

$$\begin{aligned}
 p(z_i = K^* + 1 \mid \mathbf{z}_{\setminus i}, \mathbf{s}, \mathbf{a}, \{\boldsymbol{\theta}_k\}_{k=1}^{K^*}, \alpha, \gamma) \\
 &= p(z_i = K^* + 1 \mid \mathbf{z}_{\setminus i}, \{a_t\}_{t:s_t=i}, \alpha, \gamma) \\
 &\propto p(z_i = K^* + 1 \mid \mathbf{z}_{\setminus i}, \alpha, \gamma) p(\{a_t\}_{t:s_t=i} \mid z_i = K^* + 1, \mathbf{z}_{\setminus i}, \alpha, \gamma) \\
 &\propto p(z_i = K^* + 1 \mid \mathbf{z}_{\setminus i}, \gamma) p(\{a_t\}_{t:s_t=i} \mid z_i = K^* + 1, \alpha) \\
 &\propto \gamma \cdot \int_{\Delta} p(\{a_t\}_{t:s_t=i} \mid \boldsymbol{\theta}_{K^*+1}) p_{\theta}(\boldsymbol{\theta}_{K^*+1} \mid \alpha) d\boldsymbol{\theta}_{K^*+1} \\
 &\propto \gamma \cdot \int_{\Delta} \prod_{t:s_t=i} \pi(a_t \mid \boldsymbol{\theta}_{K^*+1}) p_{\theta}(\boldsymbol{\theta}_{K^*+1} \mid \alpha) d\boldsymbol{\theta}_{K^*+1} \\
 &\propto \gamma \cdot \text{DIRMULT}(\boldsymbol{\phi}_i \mid \alpha).
 \end{aligned}$$

If a new cluster is created, we further need to initialize the corresponding control parameter $\boldsymbol{\theta}_{K^*+1}$ by performing the respective Gibbs update, i.e., by sampling from

$$\begin{aligned}
 p(\boldsymbol{\theta}_{K^*+1} \mid \mathbf{z}, \mathbf{s}, \mathbf{a}, \{\boldsymbol{\theta}_k\}_{k=1}^{K^*}, \alpha, \gamma) \\
 &= p(\boldsymbol{\theta}_{K^*+1} \mid \{a_t\}_{t:z_{s_t}=K^*+1}, \alpha) \\
 &\propto p_{\theta}(\boldsymbol{\theta}_{K^*+1} \mid \alpha) p(\{a_t\}_{t:z_{s_t}=K^*+1} \mid \boldsymbol{\theta}_{K^*+1}) \\
 &\propto p_{\theta}(\boldsymbol{\theta}_{K^*+1} \mid \alpha) \prod_{t:z_{s_t}=K^*+1} \pi(a_t \mid \boldsymbol{\theta}_{K^*+1}) \\
 &\propto \text{DIR}(\boldsymbol{\theta}_{K^*+1} \mid \boldsymbol{\xi}_{K^*+1} + \alpha \cdot \mathbf{1}^{|\mathcal{A}|}).
 \end{aligned}$$

Should a cluster get unoccupied during the sampling process, the corresponding control parameter may be removed from the stored parameter set $\{\theta_k\}$ and the index set for k needs to be updated accordingly. Note that this sampling mechanism is a specific instance of Algorithm 2 described in [Nea00]. A collapsed variant can be derived in a similar fashion.

5.2 Policy Recognition using the ddCRP

In Section 5.1, we saw that the DPMM can be derived as the nonparametric limit model of a finite mixture using a set of latent mixing proportions \mathbf{q} for the state clusters. Although the DPMM allows us to keep the number of active controllers flexible and, hence, adaptable to the complexity of the demonstration data, the CRP as the underlying clustering mechanism does not capture any spatial dependencies between the indicator variables. In fact, in the CRP, the indicators $\{z_i\}$ are coupled only via their relative frequencies (Equation 5.2) but not through their individual locations in space, yielding an *exchangeable* collection of random variables [Ald85]. Again, the spatial structure of the state space is ignored (compare prior models in Section 4.2).

The fact that DPMMs are nevertheless used for spatial clustering tasks can be explained by the particular form of data likelihood models that are used for the mixture components. In a Gaussian mixture model [Ras99], for instance, the spatial clusters emerge due to the unimodal nature of the mixture components, which encodes the locality property that is needed to obtain a meaningful spatial clustering of the data. Unfortunately, for the policy recognition problem, the DPMM is not able to exploit any spatial context via the data likelihood since the clustering is performed based on the available action information (see, for example, Equation 4.14), without direct consideration of the state information itself. The situation is particularly problematic if the expert policies overlap (i.e., when their distributions support common actions) so that the action information alone is not sufficient to discriminate between policies. For uncountable state spaces, this problem is further complicated by the fact that we observe *at most* one expert state transition per system state (compare footnote on page 51). In this case, the spatial context of the data is the only information that can resolve the ambiguity.

In order to facilitate a spatially smooth clustering, we therefore need to consider *non-exchangeable* distributions over partitionings. More specifically, as discussed in Chapters 3 and 4, we need to design our model in such a way that, whenever a state s is “close” to some other state s' and assigned to some cluster \mathcal{C}_k , then, a priori, s' should

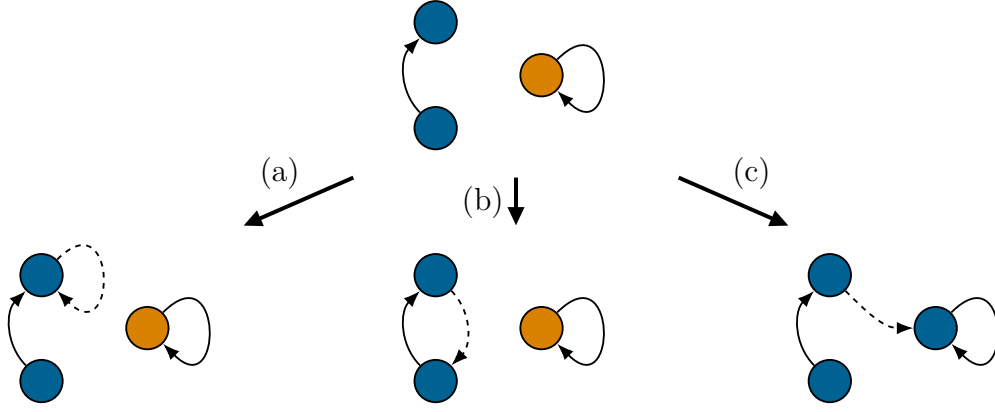


Figure 5.1: Insertion of an edge (dashed arrow) to the ddCRP graph. Colors indicate the cluster memberships of the nodes, which are defined implicitly via the connected components of the graph. **(a)** Adding a self-loop or **(b)** inserting an edge between two already connected nodes does not alter the clustering structure. **(c)** Adding an edge between two unconnected components merges the associated clusters.

belong to the same cluster \mathcal{C}_k with high probability. In that sense, we are looking for some nonparametric counterpart of the Potts model (Equation 4.19).

One model that has the desired characteristics is the distance-dependent Chinese restaurant process (ddCRP) [BF11].* As opposed to the traditional CRP, the ddCRP explicitly takes into account the spatial structure of the data: instead of assigning states to partitions, the ddCRP assigns states to other states according to their pairwise distances. These “to-state” assignments are described by a set of indicators variables $\mathbf{c} \triangleq \{c_i \in \mathcal{S}\}_{i=1}^{|\mathcal{S}|}$, where the probability that state i gets assigned to state j is defined as

$$p(c_i = j \mid \Delta, \nu) \propto \begin{cases} \nu & \text{if } i = j, \\ f(\Delta_{i,j}) & \text{otherwise.} \end{cases} \quad (5.3)$$

Herein, $\nu \in [0, \infty)$ is called the self-link parameter of the process, Δ denotes the collection of all pairwise state distances (compare Equation 4.19), and c_i is the state assignment of state i , which can be thought of as a directed edge in the graph defined on the set of all states (Figure 5.1). Accordingly, i and j take the values $\{1, \dots, |\mathcal{S}|\}$ for the finite state space model (Equation 4.11) and $\{1, \dots, T\}$ for our reduced state space

* Note that the ddCRP does *not* obey marginal invariance (see Appendix A for implications on the LfD problem). In fact, the authors of [BF11] even avoid calling this model nonparametric since it cannot necessarily be cast as a mixture model originating from a random measure. However, we stick to this term in order to make a clear distinction to the parametric models in Chapter 4, and to highlight the fact that there is no parameter that explicitly sets the number of local controllers.

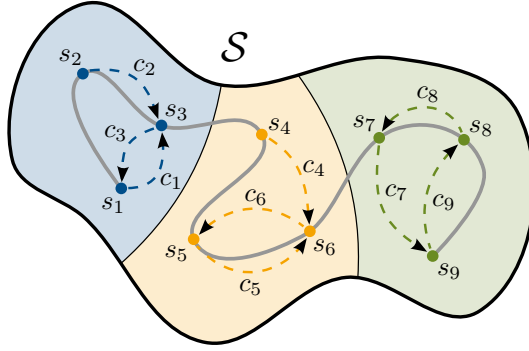


Figure 5.2: Schematic illustration of the ddCRP-based clustering applied to the reduced state space model in Section 4.3. Each trajectory state is connected to some other state of the sequence. The connected components of the resulting graph implicitly define the state clustering. Coloring of the background illustrates the spatial cluster extrapolation (see Equation A.1 in the appendix). Note that the underlying decision-making process is assumed to be discrete in time; the continuous gray line shown in the figure is only to indicate the temporal ordering of the trajectory states.

model (Equation 4.21). The state clustering is then obtained as a byproduct of this mapping via the connected components of the resulting graph (see Figure 5.1 again). In Figure 5.2, the clustering principle is illustrated for the reduced state space model.

Replacing the CRP with the ddCRP, we obtain the required conditional distribution of the state assignment c_i as

$$p(c_i = j \mid \mathbf{c}_{\setminus i}, \mathbf{s}, \mathbf{a}, \alpha, \Delta, \nu) \propto \begin{cases} \nu & \text{if } j = i, \\ f(\Delta_{i,j}) & \text{if no clusters are merged,} \\ f(\Delta_{i,j}) \cdot \mathcal{L} & \text{if clusters } \mathcal{C}_{z_i} \text{ and } \mathcal{C}_{z_j} \text{ are merged,} \end{cases}$$

where we use the shorthand notation

$$\mathcal{L} \triangleq \frac{\text{DIRMULT}(\boldsymbol{\xi}_{z_i} + \boldsymbol{\xi}_{z_j} \mid \alpha)}{\text{DIRMULT}(\boldsymbol{\xi}_{z_i} \mid \alpha) \text{DIRMULT}(\boldsymbol{\xi}_{z_j} \mid \alpha)} \quad (5.4)$$

for the data likelihood term. The derivation is analogous to that of the intention-based inference scheme described in Section 10.4.2. The statistics $\{\xi_{k,j}\}$ are defined as in Equation (4.12) but are based on the clustering that arises when we ignore the current link c_i . Accordingly, the fraction in Equation (5.4) can be interpreted as the likelihood ratio of the partitioning defined by $\mathbf{c}_{\setminus i}$ and the merged structure after inserting the new edge c_i . The resulting Gibbs sampler is a collapsed one since the local control parameters $\{\boldsymbol{\theta}_k\}$ are necessarily marginalized out during the inference process.

6

Experimental Results

In this section, we present experimental results for two types of system dynamics. As a proof of concept, we first investigate the case of uncountable state spaces, which we consider the more challenging setting for the various reasons explained in the previous chapters. To compare our framework with existing LfD methods, we further provide results for the classical Gridworld benchmark.

In order to interpret the results appropriately, it is important to keep in mind that establishing a fair comparison between LfD models is generally difficult due to their different

- working principles (intentional versus subintentional modeling),
- objectives (system identification versus optimal control),
- requirements (e.g., MDP solver, knowledge of the transition dynamics/the expert’s rationality model, countable versus uncountable state space),
- and assumptions (e.g., deterministic versus stochastic expert behavior).

With that in mind, the goal of this section is to demonstrate the prediction abilities of the considered models rather than to push the models to their individual limits. Therefore, and to reduce the overall computational load, most model hyper-parameters were manually set.

6.1 Uncountable State Space

As an illustrative example, we consider a dynamical system that describes the circular motion of an agent on a plane. The actions of the agent correspond to 24 directions that divide the space of possible angles $[0, 2\pi)$ into equally-sized intervals, i.e., action j corresponds to the angle $(j - 1)\frac{2\pi}{24}$. The transition model of the system is defined as follows: for each selected action, the agent first makes a step of length $\mu = 1$ in the intended direction. The so-obtained position is then distorted by additive zero-mean isotropic Gaussian noise of variance σ^2 . This defines the transition kernel as

$$\mathcal{T}(s_{t+1} \mid s_t, a_t = j) = \mathcal{N}(s_{t+1} \mid s_t + \mu \cdot \mathbf{e}_j, \sigma^2 \mathbf{I}), \quad (6.1)$$

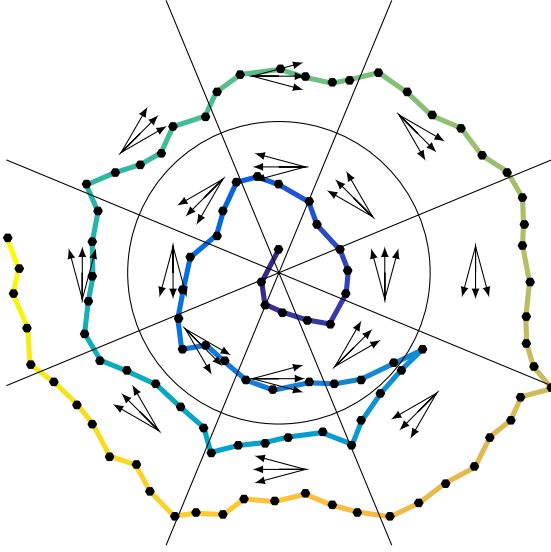


Figure 6.1: Schematic illustration of the expert policy described in Section 6.1, which applies eight local controllers to sixteen distinct regions. A sample trajectory is shown in color.

where $s_t, s_{t+1} \in \mathbb{R}^2$, \mathbf{e}_j denotes the two-dimensional unit vector pointing in the direction of action j , and \mathbf{I} is the two-dimensional identity matrix.

The intention of the agent is to describe a circular motion around the origin in the best possible manner allowed by the available actions. However, due to limited sensory information, the agent is not able to observe its exact position on the plane but can only distinguish between certain regions of the state space, as illustrated in Figure 6.1. Also, the agent is unsure about the optimal control strategy and does not always make optimal decisions but selects its actions uniformly at random from a subset of actions consisting of the optimal one and the two actions pointing to neighboring directions (see Figure 6.1 again). To increase the difficulty of the prediction task, we further let the agent change the direction of travel as soon as the critical distance of 5 to the origin is exceeded.

Based on this action strategy, we generate 10 sample trajectories of length $T = 100$. Herein, we assume a motion noise level of $\sigma = 0.2$ and initialize the agent's position uniformly at random on the unit circle. An example trajectory is depicted in Figure 6.1. The obtained trajectory data is then fed into the inference algorithms to approximate the posterior distribution over expert controllers, and the whole experiment is repeated 100 times.

For our spatial models, we use the Euclidean metric to compute the pairwise distances between states, i.e.

$$\chi(s, s') \triangleq \|s - s'\|_2. \quad (6.2)$$

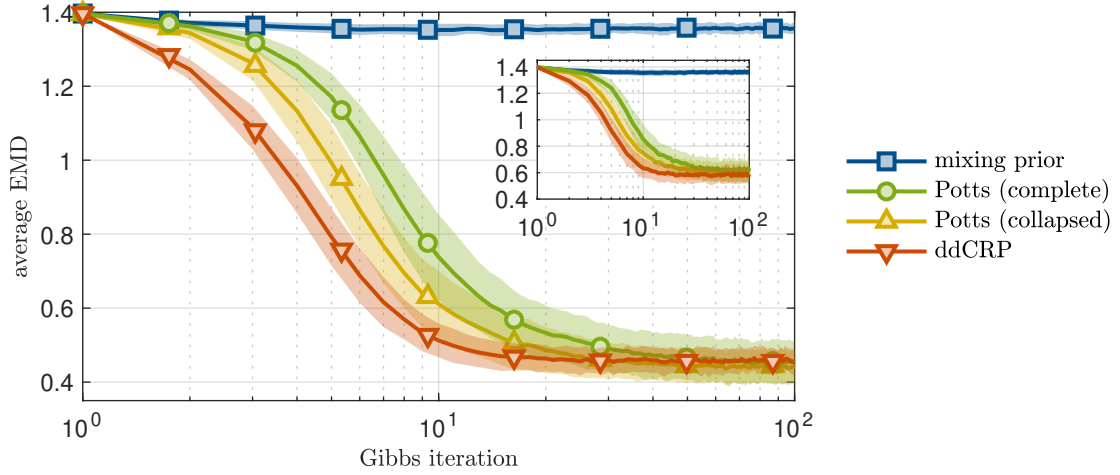


Figure 6.2: Average policy prediction error for the scenario illustrated in Figure 6.1, measured in terms of the EMD between the expert policy and the model prediction. The main figure depicts the reconstruction error at the expert trajectory states, the inset depicts the spatial prediction error at the grid states visualized in Figure 6.5. Shown are the empirical mean values and standard deviations obtained from 100 Monte Carlo runs.

The corresponding similarity values are calculated using a Gaussian-shaped kernel. More specifically, we choose

$$f_{\text{Potts}}(\Delta) \triangleq \exp\left(-\frac{\Delta^2}{\sigma_f^2}\right) \quad (6.3)$$

for the Potts model and

$$f_{\text{ddCRP}}(\Delta) \triangleq (1 - \kappa)f_{\text{Potts}}(\Delta) + \kappa$$

for the ddCRP model, with $\sigma_f = 1$ and a constant offset of $\kappa = 0.01$, which ensures that states with large distances can still join the same cluster. For the Potts model, we further use a neighborhood structure containing the eight closest trajectory points of a state. This way, each local expert policy may occur, in principle, at least once in the neighborhood of a state (recall Figure 6.1). The concentration parameter for the local controls is set to $\alpha = 1$, corresponding to a uniform prior belief over local policies.

A major drawback of the Potts model is that posterior inference about the temperature parameter β is complicated due to the nonlinear effect of the parameter on the normalization of the model. Therefore, we manually selected a temperature of $\beta = 1.6$ based on a minimization of the average policy prediction error (discussed below) via parameter sweeping. As opposed to this, we extended the inference problem for the ddCRP to the

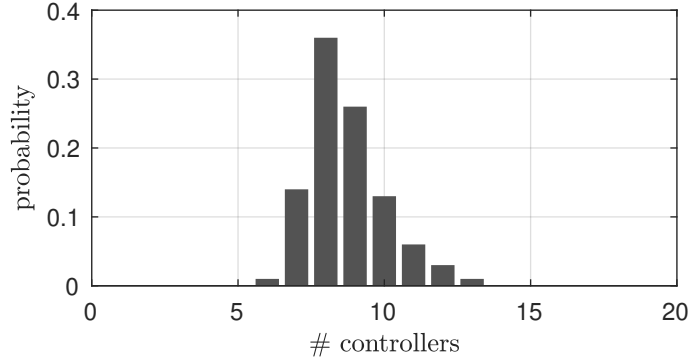


Figure 6.3: Inferred model complexity for the scenario illustrated in Figure 6.1. The figure depicts the posterior distribution of the number of local controllers used by the ddCRP model, which shows a pronounced peak at the ideal model order.

self-link parameter ν as suggested in [BF11]. For this, we used an exponential prior, i.e.

$$p_\nu(\nu) = \text{EXP}(\nu \mid \lambda),$$

with rate parameter $\lambda = 0.1$, and applied the independence Metropolis-Hastings algorithm [CG95] using $p_\nu(\nu)$ as proposal distribution with an initial value of $\nu = 1$. In all simulations, the sampler quickly converged to the stationary distribution, yielding posterior values for ν with a mean of 0.024 and a standard deviation of 0.023.

6.1.1 Parameter Inference

In order to compare the predicted policy (Equation 4.1) with the ground truth at a given query state, we compute the resulting earth mover’s distance (EMD) [RTG98] with respect to the distance metric that measures the absolute angular difference between the involved actions. To track the learning progress of all algorithms, we calculate the average EMD over all states of the given trajectory set at each Gibbs iteration. Herein, the local policy predictions are computed based on the single Gibbs sample of the respective iteration, consisting of all sampled actions, indicators and—in case of non-collapsed sampling—the local control parameters. The resulting mean EMDs and standard deviations are depicted in Figure 6.2. The inset further shows the average EMD computed on a regular state grid covering the test region depicted in Figure 6.5, which reflects the accuracy of the resulting behavior extrapolation.

As expected, the finite mixture model (using the true number of local policies and a collapsed mixing prior with $\gamma = 1$) is not able to learn a reasonable policy representation from the expert demonstrations since it does not explore the spatial structure of the data.

In fact, the resulting prediction error shows only a slight improvement as compared to an untrained model. In contrast to this, all spatial models capture the expert behavior reasonably well. In agreement with our reasoning in Section 4.1.2, we observe that the collapsed Potts model mixes significantly faster and has a smaller prediction variance than the non-collapsed version. However, the ddCRP model gives the best result, both in terms of mixing speed (see [BF11] for an explanation of this phenomenon) and model accuracy.

Interestingly, this is despite the fact that the ddCRP model additionally needs to infer the number of local controllers that are required to reproduce the expert behavior. The corresponding posterior distribution, depicted in Figure 6.3, shows indeed a pronounced peak at the true number. Moreover, the posterior sample of the local control parameters in Figure 6.4 reveals that all local policies could be identified by the model.

6.1.2 Spatial Prediction

Figure 6.5b visualizes the spatial EMD prediction errors of the trained model in the form of a heat map, which compares the ground truth policy at non-trajectory states with the mean prediction provided by our model. The test points are placed on a regular grid of size 2000×2000 centered around the origin. The required indicator variables at the grid states are computed according to Equation (A.1) in the appendix.

As to be expected, the prediction error reaches its maximum at the policy boundaries but is comparably small within each policy region, indicating a good model fit. Note that the “windmill shape” of the error can be explained as a result of the reduced state space approach in combination with the inherent asymmetry of the expert policy: regions of the state space containing trajectory endings are locally underrepresented in the data set (see example trajectory in Figure 6.1). This increases the chance of assigning the end points of a trajectory to the cluster of the preceding region, resulting in a smearing of that cluster into the next region. Also, we observe that the variance of the error (Figure 6.5c) reaches its maximum at the transition regions and generally grows with the distance to the supporting trajectory data, reflecting the increasing prediction uncertainty at cluster boundaries and regions far from the expert demonstrations.

Both described figures were computed based on the learned policy representations of 100 Monte Carlo runs. Figure 6.5a illustrates a sample partitioning obtained from one such experiment, corresponding to the inferred local controllers depicted in Figure 6.4. The found structure clearly resembles the expert partitioning in Figure 6.1.

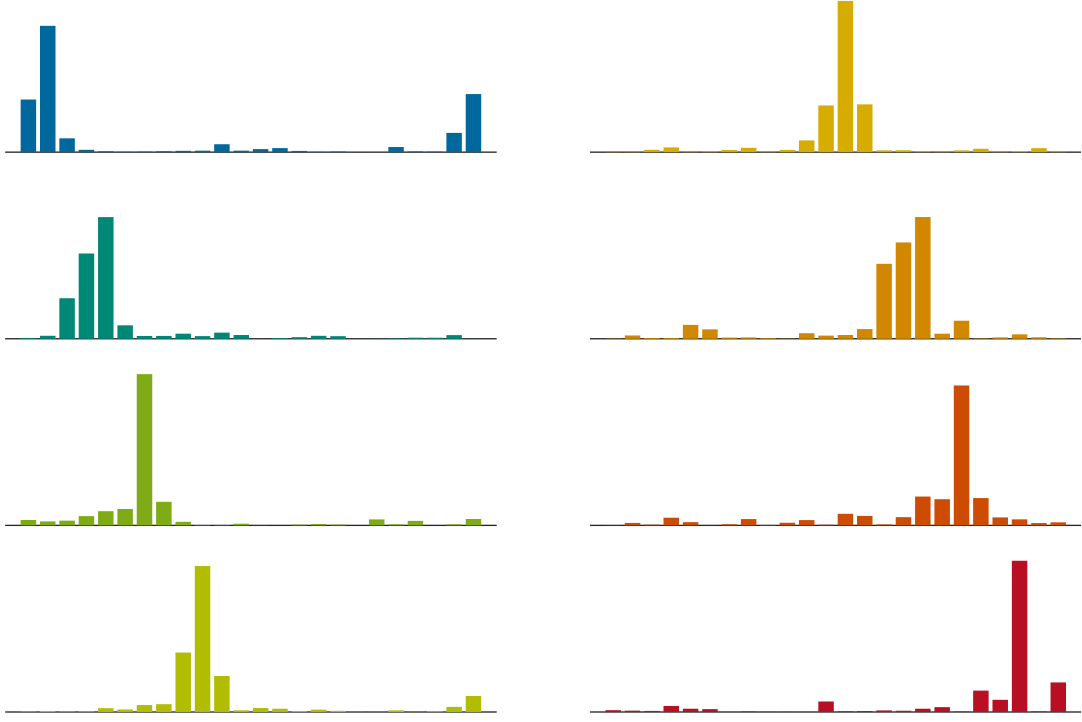


Figure 6.4: Inferred local controllers for the scenario illustrated in Figure 6.1. Each subfigure shows one of the eight local controllers obtained from a posterior sample generated by the ddCRP model, where the horizontal axis represents the action direction from 0 to 2π and the vertical axis represents probability. The spatial arrangement of these controllers for the reconstruction of the expert policy is depicted in Figure 6.5a.

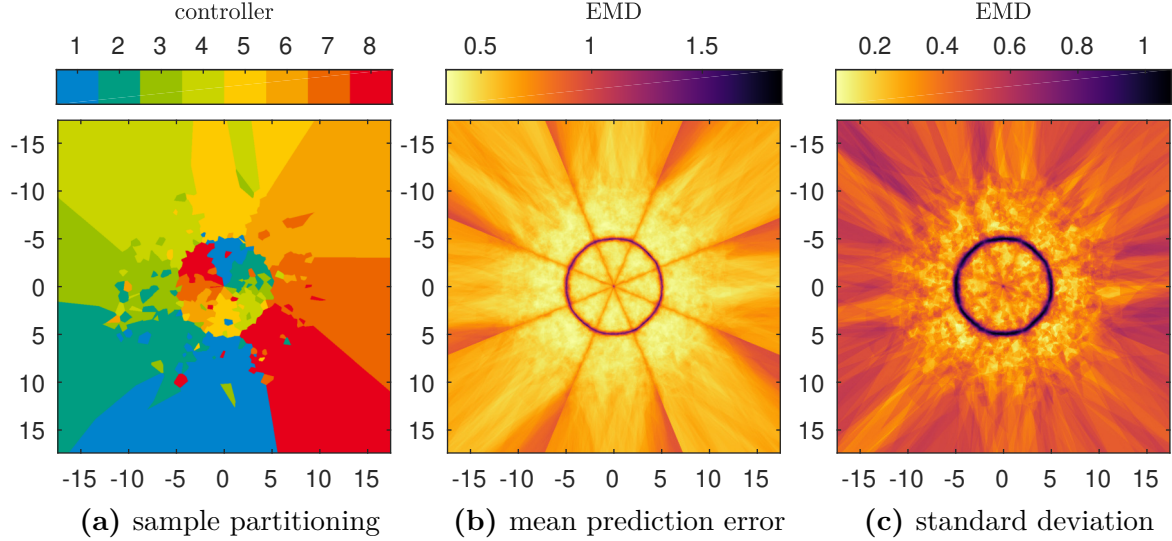


Figure 6.5: Inferred state partitioning and prediction results for the scenario illustrated in Figure 6.1. The axes indicate the location in the system state space. All figures were rendered using a spatial resolution of 2000×2000 . **(a)** Example partitioning of the state space, obtained from a posterior sample generated by the ddCRP model. Each color represents one of the local controllers depicted in Figure 6.4. **(b,c)** Empirical mean prediction error and corresponding standard deviation obtained from 100 Monte Carlo runs.

6.2 Finite State Space

In the second experiment, we test our framework against two existing LfD methods: maximum margin IRL [AN04] (max-margin) and maximum entropy IRL [Zie+08] (max-entropy). In addition, we compare the results to those obtained through maximum a posteriori expectation maximization (MAP-EM), described in Chapter 3. For the comparison, we restrict ourselves to the ddCRP model, which showed the best performance among all discussed approaches.

6.2.1 Circular Task

First, we compare all methods on a finite-size version of the circular task from Section 6.1, which is obtained by discretizing the continuous state space into a regular grid, i.e., $\mathcal{S} = \{(x, y) \in \mathbb{Z}^2 : |x|, |y| \leq 10\}$, resulting in a total of 441 states. The transition probabilities are chosen proportional to the normal densities in Equation (6.1) sampled at the grid points. For the experiment, we increase the motion noise level to $\sigma = 1$, using a reduced action set containing the eight (inter-)cardinal motion directions. Probability mass “lying outside” the finite grid area is shifted to the closest border states of the grid.

Figure 6.6a delineates the average EMD over the number of trajectories (each of length $T = 10$) provided for training. We observe that neither of the two intentional models (max-entropy and max-margin) is able to capture the demonstrated expert behavior. This is due to the fact that the circular motion of the expert cannot be explained by a simple state-dependent reward model but requires a more complex state-action reward structure, which is not considered in the original model formulations [AN04; Zie+08]. We address this issue in detail in Part II of this work. While the EM model is indeed able to capture the general trend of the data, the prediction accuracy is considerably lower than that of the ddCRP model, because it inherently assumes a deterministic behavior and thus cannot reproduce the stochastic nature of the expert policy. Expectedly, the difference in performance between the two approaches becomes even more pronounced for expert policies that distribute their probability mass on a larger subset of actions.

6.2.2 MDP Task

To analyze how the ddCRP competes against the other models in their nominal settings, we further compare all algorithms on a set of Gridworld tasks, where the expert follows MDP-optimal deterministic policies. For this purpose, we extend the system model

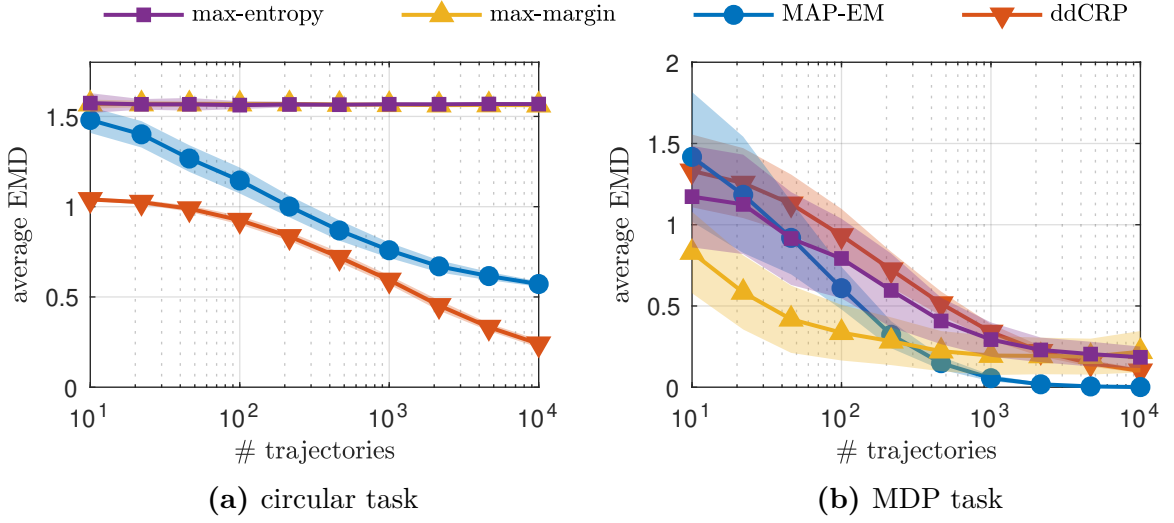


Figure 6.6: Prediction accuracies in the Gridworld setting, measured in terms of the average EMD between the expert policy and the model prediction computed over all system states. (a) Discretized version of the circular task described in Section 6.1. (b) Randomly generated MDP task. Shown are the empirical mean values and standard deviations obtained from 100 Monte Carlo runs.

from Section 6.2.1 to a proper MDP environment by adding a (randomly generated) reward mechanism for the expert. Analogous to the setting in Section 3.3, each state on the grid is assigned a reward with a probability of $\tau = 0.01$, which is then drawn from a standard normal distribution. Worlds that contain no reward are discarded. The discount factor of $\gamma = 0.9$, which is used to compute the expert policy, is provided as additional input for the intentional models (max-entropy and max-margin).

The simulation results, depicted in Figure 6.6b, show that the intentional max-margin method outperforms all other methods for small amounts of training data, whereas the subintentional methods (MAP-EM and ddCRP) yield better asymptotic estimates and smaller prediction variances. In order to interpret the shown results appropriately, we must recall that the three reference methods have a clear advantage over the ddCRP approach in this setting. First of all, the intentional methods start directly from the premise that the demonstrated behavior is the result of a preceding planning phase, which gives a head start over the subintentional approaches in this scenario. Additionally, all three reference methods assume a deterministic expert behavior a priori and do not need to infer this piece of information from the data. Despite these additional challenges, the ddCRP model yields a competitive performance.

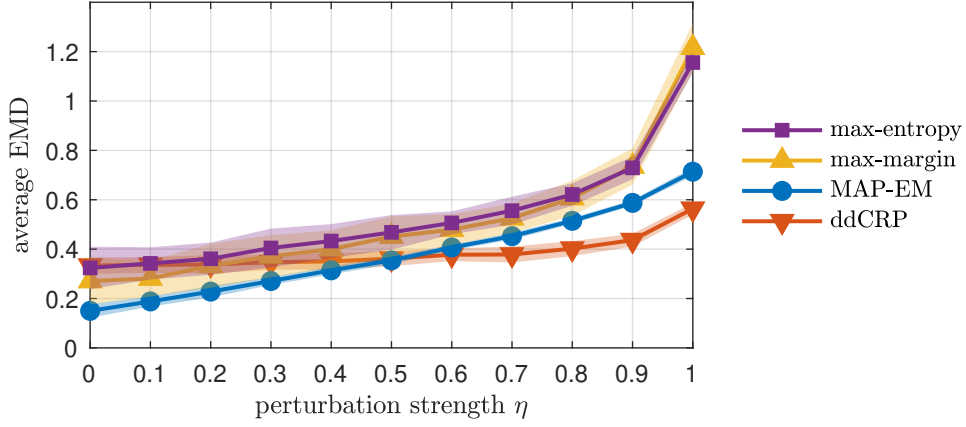


Figure 6.7: Robustness analysis with regard to modeling errors in the system dynamics. The curves represent the EMD between the true next-state distributions of the expert and those inferred by the models, averaged over the system state space. Shown are the empirical mean values and standard deviations obtained from 100 Monte Carlo runs.

6.2.3 Model Robustness

Finally, we compare all approaches in terms of their robustness to modeling errors. To this end, we repeat the experiment in Section 6.2.2 with a fixed number of 1000 trajectories but employ a different transition model for inference than used for data generation. More specifically, we assume an overly fine-grained model consisting of 24 directions, pretending that the true action set of the expert is unknown. Additionally, we perturb the model by multiplying (and renormalizing accordingly) each transition probability with a random number generated according to $f(u) = \tan(\frac{\pi}{4}(u + 1))$, where $u \sim \text{UNIFORM}(-\eta, \eta)$, for a given perturbation strength $\eta \in [0, 1]$.

Due the resulting model mismatch, a comparison to the ground truth policy based on the predicted action distribution becomes meaningless. Instead, we evaluate the accuracy of the estimated behavioral model based on the reconstruction of the predictive state dynamics. For this purpose, we compute the average EMD between the true and the predicted next-state distributions on the grid using the Euclidean distance metric, which are obtained by marginalizing the actions of the true/assumed transition model with respect to the true/learned policy, i.e.

$$p(s' | s) = \sum_{a \in \mathcal{A}} \mathcal{T}(s' | s, a) \pi(a | s).$$

Figure 6.7 depicts the resulting prediction performance for different perturbation strengths η . The graphs show that the proposed approaches are not only less susceptible

to modeling errors; also, the obtained prediction variances are notably smaller than those of the intentional models.

7

Summary

In this part of the thesis, we proposed a novel approach to policy recognition that allows to jointly learn the latent decision-making strategy of an observed expert demonstrator together with a task-appropriate representation of the system state space. With the described parametric and nonparametric models, we presented two formulations of the same problem that can be used either to learn a global system controller of prespecified complexity, or to infer the required model complexity from the observed expert behavior itself. Simulation results for both countable and uncountable state spaces and a comparison to existing frameworks demonstrated the efficacy of our approach. Most notably, the results showed that our method is able to learn accurate predictive behavioral models in situations where existing intentional methods fail, i.e., when the expert behavior cannot be explained as the result of a planning procedure. This makes our method applicable to a broader range of problems and suggests its use in a more general system identification context where no such prior knowledge about the expert behavior is available. Moreover, the task-adapted state representation learned through our framework forms the basis for further processing steps like model reduction or high-level control.

PART II

Single-agent Systems: Intentional Modeling

8 The Subgoal Principle	72
9 Bayesian Nonparametric Inverse Reinforcement Learning	76
10 Nonparametric Spatio-Temporal Subgoal Modeling	84
11 Experimental Results	102
12 Summary	115

In Part I of this work, we approached the LfD problem from a subintentional perspective and confined ourselves to reconstructing the agent’s action selection strategy, disregarding all intentional aspects of the underlying decision-making process. In this part, we take a different tack and discuss an alternative modeling paradigm based on *subgoal extraction*, which offers an intention-based approach to the LfD problem. While the new problem formulation brings an additional layer of complexity, the resulting representations provide a richer description of the observed behavior as they elicit the preferences of the demonstrator and allow to make predictions on both, the action and the intentional level.

Notation In contrast to Part I, we assume that the expert demonstrations are provided in form of separate *state-action pairs* (or state – next-state pairs) and not as a series of consecutively visited states. Although this does not change the LfD problem from a conceptual point of view, it enables a more direct comparison with prior work (see Chapter 9). Accordingly, we use the state and action subscripts to index demonstrations, not time. To be precise, the notation (s_d, a_d) refers to the d th demonstration pair. If we need to access the actual state process of the expert, we instead resort to the more explicit notation $(s_{t=1}, s_{t=2}, s_{t=3}, \dots)$. Hence, the d th demonstration pair (s_d, a_d) can be alternatively expressed as $(s_{t=t_d}, a_{t=t_d})$, where t_d denotes the corresponding decision time.

8

The Subgoal Principle

As described in Section 2.2.2, classical IRL methods such as described in [NR00; AN04; Zie+08; RA07; LPK11] assume there exists a single *global* reward model that explains the entire set of demonstrations provided by the expert. In order to relax this rather restrictive modeling assumption, recent IRL methods allow that the agent’s intention can change over time [NLJ15], or they presume that the demonstration data set is inherently composed of several parts [DR11], where different trajectories reflect the intentions of different domain experts.

In this part of the thesis, we go a step further and start from the premise that — even in the case of a single expert or trajectory — the demonstrated behavior can be explained more efficiently *locally* (i.e., within a certain context) than by a global reward model. As an illustrative example, we may consider the task shown in Figure 8.1a, where the expert approaches a set of intermediate target positions before finally heading toward a global goal state. Similarly, in Figure 8.1b, the agent eventually returns to its initial position, from where the cyclic process repeats. Despite the simplicity of these tasks, the encoding of such behaviors in a global intention model requires a reward structure that comprises a comparably large number of redundant state-action-based rewards. Alternative modeling strategies rely on task-dependent expansions of the agent’s state representation, e.g., to memorize the last visited goal [Kri+16], or they resort to more general decision-making frameworks like semi-MDPs/options [BD94; SPS99] in order to achieve the necessary level of task abstraction.

In the following sections, we present a substantially simpler modeling framework that requires only minimal adaptations to the standard MDP formalism but comes with a hypothesis space of behavioral models that is sufficiently large to cover a broad class of expert policies. The key insight that motivates our approach is that many tasks, like those in Figure 8.1, can be decomposed into smaller subtasks that require considerably less modeling effort. The resulting low-level task descriptions can then be used as building blocks to synthesize arbitrarily complex behavioral strategies through a suitable sequencing of subtasks. This offers the possibility to learn comparably simple task representations using the intuitive concept of *subgoals*, which is achieved by efficiently encoding the expert behavior using task-adapted partitionings of the system state space/the expert data.

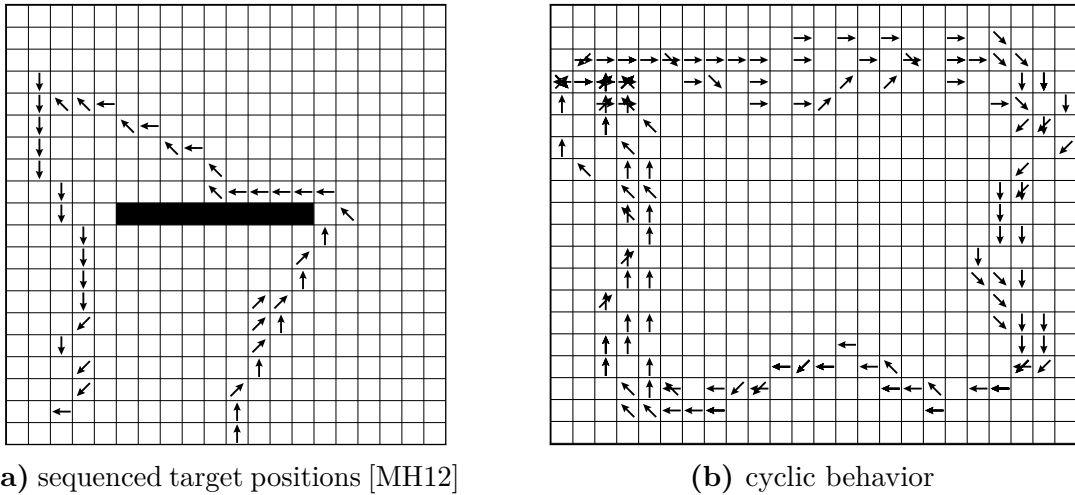


Figure 8.1: Two simple behavior examples that motivate the subgoal principle. The setting is based on the Gridworld dynamics described in Section 11.1. In both cases, a task description based on a global reward function is inefficient as it requires many state-action-based rewards to explain the observed trajectory structures. However, the data can be described efficiently through subgoal-based encodings. Both scenarios are analyzed in detail in Chapter 11.

The proposed framework builds upon the method of Bayesian nonparametric inverse reinforcement learning (BNIRL) [MH12], which can be used to build a subgoal representation of a task based on demonstration data — however, without learning the underlying subgoal relationships or providing a policy model that can generalize the strategy of the demonstrator. In order to address this limitation, we generalize the BNIRL model using the insights we have gathered in Part I, building a compact intentional model of the expert’s behavior that explicitly describes the local dependencies between the demonstrations and the underlying subgoal structure. The result is an integrated Bayesian prediction framework that exploits the spatio-temporal context of the demonstrations and is capable of producing smooth policy estimates that are consistent with the expert’s plan.

Related Work

The idea of decomposing complex behavior into smaller parts has been around for long and researchers have approached the problem in many different ways. Because the overall field of methods is too large to be covered here, we restrict ourselves to demonstration-based approaches, with a focus on intentional methods. For an overview of reinforcement-learning-based methods, we refer to existing literature, e.g., the work by Daniel et al. [Dan+16] and the survey by Argall et al. [Arg+09].

First, there is the class of methods that pursue a decomposition of the observed behavior on the global level, using trajectory-based IRL approaches. For example, Dimitrakakis and Rothkopf [DR11] proposed a hierarchical prior over reward functions to account for the fact that different trajectories in a data set could reflect different behavioral intentions, e.g., because they were generated by different domain experts. Similarly, Babeş-Vroman et al. [Bab+11] follow an expectation-maximization-based clustering approach to group individual trajectories according to their underlying reward functions. Choi and Kim [CK12] generalized this idea by proposing a nonparametric Bayesian model in which the number of intentions is a priori unbounded.

While the above methods consider the expert data at a global scale, our work is concerned with the problem of subgoal modeling, which is often conducted in the form of option-based reasoning [SPS99]. For instance, Tamassia et al. [Tam+15] proposed a clustering approach based on state distances to find a minimal set of options that can explain the expert behavior. While the method provides a simple alternative to handcrafting options, it does not allow any probabilistic treatment of the data and involves many ad-hoc design choices. Going in the same direction, Daniel et al. [Dan+16] presented a more principled, probabilistic option framework based on expectation maximization. Not only is the framework capable of inferring sub-policies automatically, it can be also used in a reinforcement learning context for intra-option learning. However, the resulting behavioral model is based on point estimates of the policy parameters, and the number of sub-policies needs to be specified manually. The latter problem was solved by Krishnan et al. [Kri+16], who proposed a hierarchical nonparametric IRL framework to learn a sequential representation of the demonstrated task, based on a set of transition regions that are defined through local changes in linearity of the observed behavior. However, in contrast to the work by Daniel et al. [Dan+16], inference is not performed jointly but in several isolated stages where, again, each stage only propagates a point estimate of the associated model parameters. Moreover, the temporal relationship of the demonstration data, used to identify the local linearity changes, is considered only in an ad-hoc fashion with the help of a windowing function.

Another general class of models, which explicitly addresses this issue, employs a hidden Markov model (HMM) structure to establish a temporal relationship between the demonstrations. For instance, the work presented by Nguyen et al. [NLJ15] can be regarded as a generalization of the model by Babeş-Vroman et al. [Bab+11], which extends the expectation maximization framework by imposing a Markov structure on the reward model. Similarly, Niekum et al. [Nie+12] use an extended HMM to segment

the demonstrations into vector autoregressive models, in order to learn a suitable set of movement primitives [Sch+05]. However, the learning of those primitives is done in a post-processing step, meaning that the quality of the final representation crucially depends on the success of the initial segmentation stage. In contrast, the method by Rückert et al. [Rüc+13] automatically learns the position and timing of subgoals in the form of via-points, but the number of via-points is assumed to be known and the system objective gets finally encoded in form of a global cost function. Recently, Lioutikov et al. [Lio+17] presented a related approach based on probabilistic movement primitives that jointly solves the segmentation and learning step for an unknown number of primitives, using an expectation maximization framework. Yet, the model operates purely on the trajectory level and cannot reveal the latent intentions of the demonstrator. Another variant of the approach by Niekum et al. [Nie+12] that explicitly addresses this problem was proposed by Surana and Srivastava [SS14]. In their paper, the authors propose to replace the HMM emission model with an MDP model, in order to infer a policy model from the segmented trajectories instead of recognizing changes in the dynamics. The model was later extended by Ranchod et al. [RRK15], who augmented the HMM representation with a beta process model to facilitate skill sharing across trajectories. While the resulting model formulation is highly flexible, its major drawback is that inference becomes computationally expensive as it involves multiple IRL iterations per Gibbs step.

In contrast to the HMM-based solutions, which by their sequential nature focus on the *temporal* relationship of subtasks, the approach presented in this thesis establishes a more general correlation structure between demonstrations by employing non-exchangeable prior distributions over subgoal assignments, i.e., without committing to purely temporal factorizations of subgoals. This results in a compact model representation (e.g., it avoids the need of estimating latent subgoal transition probabilities required in an HMM structure) and adds the flexibility to capture both, the temporal and the spatial dependencies between subtasks.

9

Bayesian Nonparametric Inverse Reinforcement Learning

The purpose of this section is to recapitulate the principle of Bayesian nonparametric inverse reinforcement learning [MH12], which lays the foundation for the contributions in this part. After briefly discussing all building blocks of the model, we focus on the limitations of the framework, which motivates the need for an extended model formulation and finally leads to a new inference approach, presented afterwards in Chapter 10.

9.1 Revisiting the BNIRL Framework

Following the common IRL paradigm [NR00; ZJ12], the goal of BNIRL is to infer the intentions of an agent based on demonstration data. Starting from a CMP model (Chapter 3), the problem is formalized on a finite state space \mathcal{S} assuming a time-invariant state transition model $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, where \mathcal{A} is a finite set of actions available to the agent at each state. For notational convenience, we represent the states in \mathcal{S} by the integer values $\{1, \dots, |\mathcal{S}|\}$, where $|\mathcal{S}|$ denotes the cardinality of the state space.

In BNIRL, it is assumed that we can observe a number of expert demonstrations provided in the form of state-action pairs, $\mathcal{D} \triangleq \{(s_d, a_d)\}_{d=1}^D$, where each pair $(s_d, a_d) \in \mathcal{S} \times \mathcal{A}$ consists of a state s_d visited by the agent and the corresponding action a_d taken (see page 71). Herein, D denotes the size of the demonstration set. Throughout the rest of this part, we will use the shorthand notations $\mathbf{s} \triangleq \{s_d\}_{d=1}^D$ and $\mathbf{a} \triangleq \{a_d\}_{d=1}^D$ to access the collections of expert states and actions individually. Note that the BNIRL model makes no assumptions about the temporal ordering of the demonstrations, i.e., each state-action pair is considered to have arisen from a specific but arbitrary time instant of the agent's decision-making process. We will come back to this point later in Sections 9.2 and 10.3.

In contrast to the classical MDP formalism and most other IRL frameworks, BNIRL does *not* presuppose that the observed expert behavior necessarily originates from a single underlying reward function. Instead, it introduces the concept of *subgoals* (and corresponding *subgoal assignments*) with the underlying assumption that, at each decision instant, the expert selects a particular subgoal to plan the next action. Each subgoal is herein represented by a certain reward function defined on the system state

space; in the simplest case, it corresponds to a single reward mass placed at a particular goal state in \mathcal{S} , which we identify with a reward function $R_g : \mathcal{S} \rightarrow \{0, C\}$ of the form

$$R_g(s) \triangleq \begin{cases} C & \text{if } g = s, \\ 0 & \text{otherwise,} \end{cases} \quad (9.1)$$

where $g \in \{1, \dots, |\mathcal{S}|\}$ indicates the subgoal location and $C \in (0, \infty)$ is some positive constant (compare [SWB05; SP02; Tam+15]).

Although in principle it is legitimate to associate each subgoal with an arbitrary reward structure to encode more complex forms of goal-oriented behavior (see, for example, [RRK15]), the restriction to the reward function class in Equation (9.1) is sufficient in the sense that the same behavioral complexity can be synthesized through a combination of subgoals. This is made possible by the nonparametric nature of BNIRL, i.e., because the number of possible subgoals is assumed to be unbounded. The use of the reward model in Equation (9.1) has the advantage, however, that posterior inference about the expert's subgoals becomes computationally tractable, as will be explained in Section 10.4.5. In the following, we therefore focus on the above reward model and summarize the infinite collection of subgoals in the multiset $\mathcal{G} \triangleq \{g_k\}_{k=1}^{\infty} \in \times_{k=1}^{\infty} \mathcal{S}$, where we adopt the assumption that $p(\mathcal{G} | \mathbf{s}) = \prod_{k=1}^{\infty} p_g(g_k | \mathbf{s})$.*

The subgoal assignment in BNIRL is achieved using a set of indicator variables $\tilde{\mathbf{z}} \triangleq \{\tilde{z}_d \in \mathbb{N}\}_{d=1}^D$, which annotate each demonstration pair (s_d, a_d) with its unique subgoal index. Similar to the DPMM formulation in Section 5.1, the prior distribution $p(\tilde{\mathbf{z}})$ is modeled by a CRP, which assigns the event that indicator \tilde{z}_d points to the j th subgoal the prior probability

$$p(\tilde{z}_d = j | \tilde{\mathbf{z}}_{\setminus d}) \propto \begin{cases} n_j & \text{if } j \in \{1, \dots, K\}, \\ \alpha & \text{if } j = K + 1, \end{cases}$$

where $\tilde{\mathbf{z}}_{\setminus d} \triangleq \{\tilde{z}_d\} \setminus \tilde{z}_d$ is a shorthand notation for the collection of all indicator variables except \tilde{z}_d . Further, n_j denotes the number of assignments to the j th subgoal in $\tilde{\mathbf{z}}_{\setminus d}$, K represents the number of distinct entries in $\tilde{\mathbf{z}}_{\setminus d}$, and $\alpha \in [0, \infty)$ is a parameter controlling the diversity of assignments.

* Notice that the subgoal prior distribution in the original BNIRL formulation [MH12] does not take the state variable \mathbf{s} as an argument. Nonetheless, the authors of BNIRL suggest to restrict the support of the distribution to the set of visited states, which indeed implies a conditioning on \mathbf{s} .

Having targeted a particular subgoal g_{z_d} while being at some state s_d , the expert is assumed to choose the next action a_d according to a softmax decision rule, $\pi : \mathcal{A} \times \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$, which weighs the expected returns of all actions against one another, i.e.

$$\pi(a_d | s_d, g_{z_d}) \triangleq \frac{\exp \left\{ \beta Q^*(s_d, a_d | g_{z_d}) \right\}}{\sum_{a \in \mathcal{A}} \exp \left\{ \beta Q^*(s_d, a | g_{z_d}) \right\}}. \quad (9.2)$$

Herein, $Q^*(s, a | g)$ denotes the state-action value of action a at state s under an optimal policy for the subgoal reward function R_g , i.e.

$$Q^*(s, a | g) \triangleq \max_{\bar{\pi}} \mathbb{E} \left[\sum_{n=0}^{\infty} \gamma^n R_g(s_{t=n}) \mid s_{t=0} = s, a_{t=0} = a, \bar{\pi} \right], \quad (9.3)$$

where the expectation is with respect to the stochastic state-action sequence induced by the fixed policy $\bar{\pi} : \mathcal{S} \rightarrow \mathcal{A}$, with initial action a executed at the starting state s .

The softmax policy π models the expert's (in-)ability to maximize the future expected return in view of the targeted subgoal, while the coefficient $\beta \in [0, \infty)$ is used to express the expert's level of confidence in the optimal action. Combined with the subgoal prior distribution p_g and the partitioning model $p(\tilde{\mathbf{z}})$, we obtain the joint distribution of all demonstrated actions \mathbf{a} , subgoals \mathcal{G} , and subgoal assignments $\tilde{\mathbf{z}}$ as

$$p(\mathbf{a}, \tilde{\mathbf{z}}, \mathcal{G} | \mathbf{s}) = p(\tilde{\mathbf{z}}) \prod_{k=1}^{\infty} p_g(g_k | \mathbf{s}) \prod_{d=1}^D \pi(a_d | s_d, g_{z_d}). \quad (9.4)$$

The structure of this distribution is visualized in form of a Bayesian network in Figure 10.2a. It is worth emphasizing that π — although referred to as the likelihood model for the state-action *pairs* in the original BNIRL paper — is really just a model for the actions *conditional on the states*. In contrast to what is stated in the original paper, the distribution in Equation (9.4) therefore takes the form of a conditional distribution (i.e., conditional on \mathbf{s}), which does not provide any generative model for the state variables.

Posterior inference in BNIRL refers to the (approximate) computation of the conditional distribution $p(\tilde{\mathbf{z}}, \mathcal{G} | \mathcal{D})$, which allows to identify potential subgoal locations and the corresponding subgoal assignments based on the available demonstration data. For further details, the reader is referred to the original paper [MH12].

9.2 Limitations of BNIRL

Subgoal-based inference is a well-motivated approach to IRL and the BNIRL framework has shown promising results in a variety of real-world scenarios. Yet, the model formulation by Michini and How [MH12] comes with a number of significant conceptual limitations, which we explain in detail in the following paragraphs.

Limitation 1:

Subgoal Exchangeability and Posterior Predictive Policy

The central limitation of BNIRL is that the framework is restricted to pure subgoal extraction and does *not* inherently provide a reasonable mechanism to generalize the expert behavior based on the inferred subgoals. The reason lies in the particular design of the framework, which, at its heart, treats the subgoal assignments $\tilde{\mathbf{z}}$ as *exchangeable random variables* (compare Section 5.2). By implication, the induced partitioning model $p(\tilde{\mathbf{z}})$ is agnostic about the covariate information contained in the data set and the resulting behavioral model is unable to propagate the expert knowledge to new situations.

To illustrate the problem, let us investigate the predictive action distribution that arises from the original BNIRL formulation. For simplicity and without loss of generality, we may assume that we have perfectly inferred all subgoals \mathcal{G} and corresponding subgoal assignments $\tilde{\mathbf{z}}$ from the demonstration set \mathcal{D} . Denoting by $a^* \in \mathcal{A}$ the predicted action at some new state $s^* \in \mathcal{S}$, the BNIRL model yields

$$\begin{aligned}
 p(a^* | s^*, \mathcal{D}, \tilde{\mathbf{z}}, \mathcal{G}) &= \sum_{\tilde{z}^* \in \mathbb{N}} p(a^*, \tilde{z}^* | s^*, \mathcal{D}, \tilde{\mathbf{z}}, \mathcal{G}) \\
 &= \sum_{\tilde{z}^* \in \mathbb{N}} p(a^* | \tilde{z}^*, s^*, \mathcal{D}, \tilde{\mathbf{z}}, \mathcal{G}) p(\tilde{z}^* | s^*, \mathcal{D}, \tilde{\mathbf{z}}, \mathcal{G}) \\
 &\stackrel{(\star)}{=} \sum_{\tilde{z}^* \in \mathbb{N}} p(a^* | \tilde{z}^*, s^*, g_{\tilde{z}^*}) p(\tilde{z}^* | \tilde{\mathbf{z}}),
 \end{aligned} \tag{9.5}$$

where $\tilde{z}^* \in \mathbb{N}$ is the latent subgoal index belonging to s^* . Note that $p(a^* | \tilde{z}^*, s^*, g_{\tilde{z}^*})$ can either represent the softmax decision rule $\pi(a^* | s^*, g_{\tilde{z}^*})$ from Equation (9.2) or an optimal (deterministic) policy for subgoal $g_{\tilde{z}^*}$, depending on whether we aspire to describe the *noisy expert behavior* at s^* or want to determine an *optimal action* according to the inferred reward model. The last equality in Equation (9.5), indicated by (\star) , follows from the conditional independence properties implied by Equation (9.4), which can be easily verified using d-separation [KF09] on the graphical model in Figure 10.2a.

As Equation (9.5) reveals, the predictive model is characterized by the posterior distribution $p(\tilde{z}^* | s^*, \mathcal{D}, \tilde{\mathbf{z}}, \mathcal{G})$ of the latent subgoal assignment \tilde{z}^* of state s^* — the intuition being that, in order to generalize the expert’s plan to a new situation, we need to take into account the gathered information about what would be a likely subgoal targeted by the expert at s^* . However, in BNIRL, the distribution $p(\tilde{z}^* | s^*, \mathcal{D}, \tilde{\mathbf{z}}, \mathcal{G})$ is modeled *without consideration of the query state s^* or any other observed variable*. By conditional independence (Equation 9.4), the distribution effectively reduces (\star) to the CRP prior $p(\tilde{z}^* | \tilde{\mathbf{z}})$, which, due to its intrinsic exchangeability property, only considers the subgoal frequencies of the readily inferred assignments $\tilde{\mathbf{z}}$. Clearly, a subgoal assignment mechanism based solely on frequency information is of little use when it comes to predicting the expert behavior as it will inevitably ignore the structural information contained in the demonstration set and always return the same subgoal probabilities at all query states, regardless of the agent’s actual situation. By contrast, a reasonable assignment mechanism should inherently take into account the context of the agent’s current state s^* when deciding about the next action.

While the authors of BNIRL discuss the action selection problem in their paper and propose an assignment strategy for new states based on action marginalization, their approach does not provide a satisfactory solution to the problem because the alleged conditioning on the query state (see Equation 19 in the original paper [MH12]) has no effect on the involved subgoal indicator variable, as shown by Equation (9.5) above. The only way to remedy the problem *without* modifying the model is to use an external post-processing scheme like the waypoint method, discussed in the next section.

Limitation 2: Spatial and Temporal Context

The waypoint method, described at full length in a follow-up paper by Michini et al. [Mic+15], is a post-processing routine to convert the subgoals identified through BNIRL into a valid option model [SPS99]. The obtained model reconstructs the high-level plan of the demonstrator by sequencing the inferred subgoals in a way that complies with the spatio-temporal relationships of the expert’s decisions as observed during the demonstration phase. To this end, the required initiation and termination sets of the option-policies are constructed by considering the state distances to the identified subgoals as well as their temporal ordering prescribed by the expert.

When combined with BNIRL, this method allows to synthesize a behavioral model that mimics the observed expert behavior. However, the strategy comes with a number of significant drawbacks:

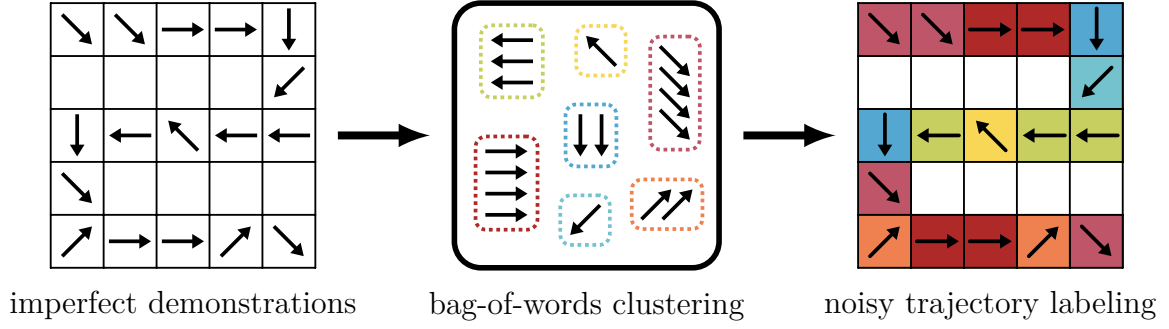


Figure 9.1: A diagram to illustrate the implications of the exchangeability assumption in BNIRL. Similar to a *bag-of-words model* [BNJ03; Yan+07], the BNIRL partitioning mechanism ignores the spatio-temporal context of the data, which makes it difficult to discriminate demonstration noise from a real change of the agent’s intentions. Note that the diagram illustrates the partitioning process in a simplified way as it only shows the effect of the prior $p(\tilde{\mathbf{z}})$ but neglects the impact of the likelihood model π . While the latter *does* indeed consider the state context of the actions, it cannot account for spatial or temporal patterns in the data as it processes all state-action pairs separately.

- (i) Using the waypoint method, the spatio-temporal relationships between the individual demonstrations are explored only in a post-hoc fashion and are largely ignored during the actual inference procedure (the state information enters via the likelihood model π but is not considered by the partitioning model $p(\tilde{\mathbf{z}})$, as explained in Limitation 1). This lack of context-awareness makes the inference mechanism overly prone to demonstration noise (see Figure 9.1 and results in Chapter 11).
- (ii) Measuring proximities to subgoals in order to determine the right visitation order requires some form of distance metric defined on the state space. If the system states correspond to physical locations, constructing such a metric is usually straightforward. However, in the general case where states encode arbitrary abstract information (see example in Section 11.2), it can become difficult to design that metric by hand. Unfortunately, the BNIRL framework does not provide any solution to this problem.
- (iii) The waypoint method cannot be applied to multiple unaligned trajectories (e.g., obtained from different experts) or in cases where the data set does not carry any temporal information. This situation occurs, for instance, when the expert data is provided as separate state-action pairs with unknown timestamps and not given in form of coherent trajectories (see again example in Section 11.2).
- (iv) Assigning a particular visitation order to the inferred subgoals is meaningful only if the expert eventually reaches those subgoals during the demonstration

phase (or if, at least, the subgoals lie “close” to the visited states in terms of the aforementioned distance metric). Finding subgoals with such properties can be guaranteed by constraining the support of the subgoal prior distribution p_g to states that are near to the expert data (see footnote on page 77) but this reduces the flexibility of the model and potentially disables compact encodings of the task (Figure 9.2).

Limitation 3: Inconsistency under Time-Invariance

Reasoning about the intentions of an agent, there are two basic types of behavior one may encounter:

- either the agent follows a static strategy to optimize a fixed objective (as assumed in the standard MDP formalism, Chapter 2), or
- the intentions of the agent change over time.

The latter is clearly the more general case but also poses a more difficult inference problem in that it requires us both, to identify the intentions of the agent and to understand their temporal relationship. The static scenario, in contrast, implies that there exists an optimal policy for the task in form of a simple state-to-action mapping $\pi : \mathcal{S} \rightarrow \mathcal{A}$ (Section 2.1.5), which from the very beginning imprints a specific structure on the inference problem.

The BNIRL model generally falls into the second category since it freely allocates its subgoals per *decision instant* and not per state, allowing a flexible change of the agent’s objective. Yet, it is important to understand that the model does not actually distinguish between the two described scenarios. As explained in Limitation 2, the temporal aspect of the data is not explicitly modeled by the BNIRL framework, even though the waypoint method subsequently tries to capture the overall chronological order of events. As a consequence, the model is not tailored to either of the two scenarios: on the one hand, it ignores the valuable temporal context that is needed in the time-varying case to reliably discriminate demonstration noise from a real change of the agent’s intention (Figure 9.1). On the other hand, the model is agnostic about the predefined time-invariant nature of the optimal policy in the static scenario. This lack of structure not only makes the inference problem harder than necessary in both cases; it also allows the model to learn inconsistent data representations in the static case since the same state can be potentially assigned to more than one subgoal, violating the above-mentioned state-to-action rule (Figure 9.3).

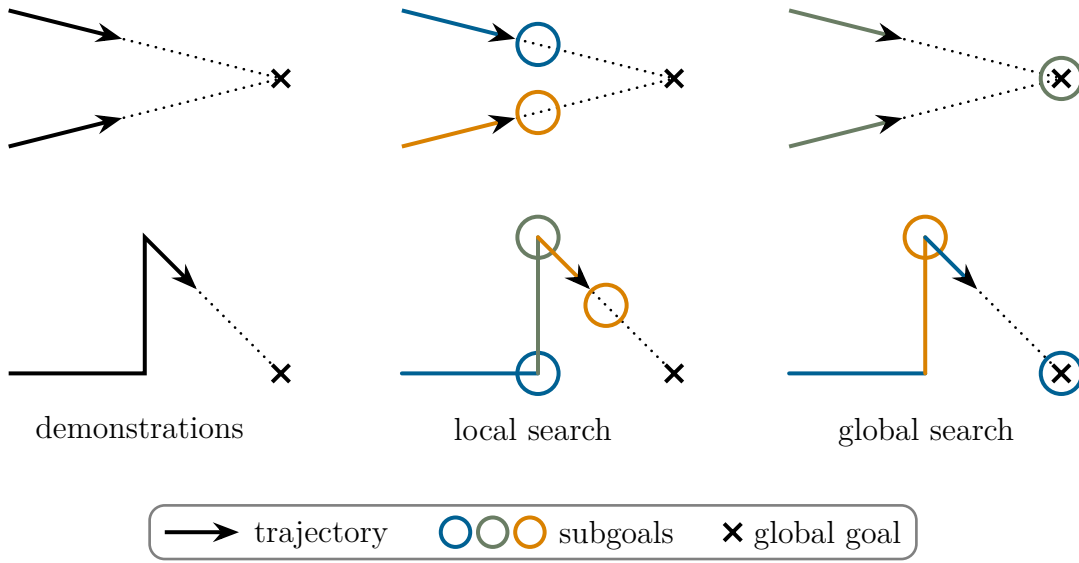


Figure 9.2: Difference between local (constrained) and global (unconstrained) subgoal search. The top and the bottom row depict two different sets of demonstration data (solid lines), together with potential goal/subgoal locations (crosses/circles) that explain the observed behavior. Color indicates the corresponding subgoal assignment of each trajectory segment. **Top:** Two trajectories approaching the same goal. **Bottom:** The agent is heading toward a global goal, gets temporarily distracted, and then follows up on its original plan. **Left:** Observed trajectories. **Center:** Example partitioning under the assumption that the expert reached all subgoals during the demonstration. **Right:** Example partitioning without restriction on the subgoal locations, yielding a more compact encoding of the task.

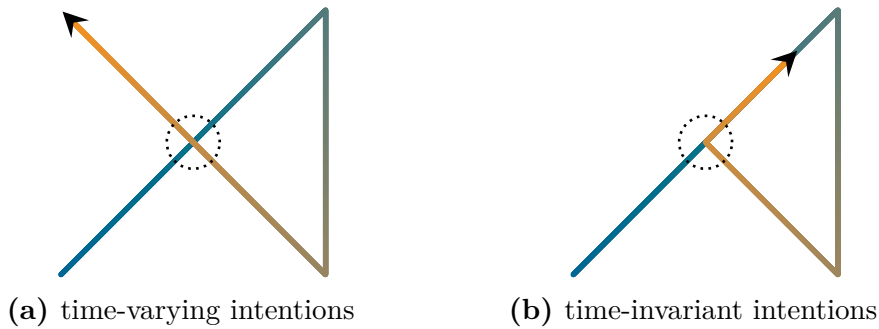


Figure 9.3: Schematic comparison of the two basic behavior types, illustrated using two different agent trajectories. Color indicates the temporal progress. **(a)** Time-varying intentions may cause the agent to perform a different action when revisiting a state (dotted circle). **(b)** By contrast, time-invariant intentions imply a simple state-to-action policy: the agent has no incentive to perform a different action at an already visited state since—by definition—the underlying objective remained unchanged. Diverging actions, as observed at the crossing point in the left subfigure, can therefore only be explained as a result of suboptimal behavior.

Limitation 4: Subgoal Likelihood Model

Apart from the discussed limitations of the BNIRL partitioning model, it turns out there are two problematic issues concerning the softmax likelihood model in Equation (9.2). On the following pages, we demonstrate that the specific form of the model encodes a number of properties that are indeed contradictory to our intuitive understanding of subgoals. While these properties are less critical for the final prediction of the expert behavior, it turns out they drastically affect the *localization* of subgoals. Since the cause of these effects is somewhat hidden in the model equation, we defer the detailed explanation to Section 10.1.

Limitation 5: State-Action Demonstrations

Lastly, a minor problem of the original BNIRL framework is that the inference algorithm expects the demonstration data to be provided in the form of state-action pairs, which requires full access to the expert’s action record. This assumption is restrictive from a practical point of view as it confines the application of the model to settings with laboratory-like conditions that allow a complete monitoring of the expert. For this reason, it is important to note that an estimate of the expert’s action sequence can be recovered through BNIRL with the help of an additional sampling stage (omitted in the original paper [MH12]), provided that we know the successor state reached by the expert after each decision. For the marginalized inference scheme described in this work, we present the corresponding sampling stage in Section 10.4.4.

10

Nonparametric Spatio-Temporal Subgoal Modeling

In this section, we introduce a redesigned inference framework, which, in analogy to BNIRL, we refer to as distance-dependent Bayesian nonparametric IRL (ddBNIRL). We derive the model by making a series of modifications to the original BNIRL framework that address the previously described shortcomings on the conceptual level. Rethinking each part of the original framework, we begin with a discussion of the commonly used softmax action selection strategy (Equation 9.2) in the context of subgoal inference, which finally leads to a redesign of the subgoal likelihood model (Limitation 4). Next, we

focus on the subgoal allocation mechanism itself and introduce two closely related model formulations, each targeting one of the basic behavior types described in Figure 9.3, thereby addressing Limitations 1, 2 and 3. For the time-invariant case, we begin with an intermediate model that introduces a subtle yet important structural modification to the BNIRL framework. In a second step, we generalize that new model to account for the spatial structure of the control problem, which finally allows us to extrapolate the expert behavior to unseen situations. As part of this generalization, we present a new state space metric that arises naturally in the context of subgoal inference (see Limitation 2, second point). Lastly, we tackle the time-varying case and present a variant of the model that explicitly considers the temporal aspect of the subgoal problem. A solution to Limitation 5 is discussed later in Section 10.4.

In contrast to BNIRL, both presented models can be used likewise for *subgoal extraction* and *action prediction*. Moreover, sticking with the Bayesian methodology, the presented approach provides complete posterior information at all levels.

10.1 The Subgoal Likelihood Model

Like many other approaches found in the IRL literature, BNIRL exploits a softmax weighting (Equation 9.2) to transform the Q-values of an optimal policy into a valid subgoal likelihood model. The softmax action rule has its origin in reinforcement learning, where it is known as the Boltzmann exploration strategy [Ces+17; SB98], which is commonly applied to cope with the *exploration-exploitation dilemma* [Gha+15]. In recent years, however, it has also become the de facto standard for describing the (imperfect) decision-making strategy of an observed demonstrator (see, for example, [DR11; RA07; RD11; CK12; NS07; Bab+11]).

In the following paragraphs, we focus on the implications of this model on the subgoal extraction problem and show that it contradicts our intuitive understanding of what characteristics a reasonable subgoal model should have. In particular, we argue that the subgoal posterior distribution arising from the BNIRL softmax model is of limited use for inferring the latent intention of the agent, due to subgoal artifacts caused by the system dynamics that cannot be reconciled with the evidence provided by the demonstrations. Based on these insights, we propose an alternative transformation scheme that is more consistent with the subgoal principle.

10.1.1 Scale of the Reward Function

The first implication of the softmax likelihood model concerns the choice of the uncertainty coefficient β . To explain the problem, we consider the thought experiment of an agent located at some state s targeting a particular subgoal g . The likelihood $\pi(a | s, g)$ in Equation (9.2) quantifies the probability that the agent decides for a specific action a , based on the corresponding state-action values $Q(\cdot, s | g)$. Since those values are linear in the underlying reward function R_g (Equation 9.3), the softmax likelihood model implies that the expert's ability to maximize the long-term reward, reflected by the spread of the probability mass in $\pi(\cdot | s, g)$, rises with the magnitude C of the assumed subgoal reward (more concentrated probability mass signifies a higher confidence in the action choice). In other words, assuming a higher goal reward virtually increases our level of confidence in the expert, even though the difficulty of the underlying task and the optimal policy remain unchanged. Nonetheless, the BNIRL model requires us to readjust the uncertainty coefficient β in order to keep both models consistent. However, as the model provides no reference level for the expert's uncertainty across different scenarios, the choice of β becomes nontrivial. Yet, the parameter has a significant impact on the granularity of the learned subgoal model as it trades off purposeful goal-oriented behavior against random decisions.

Note that the described effect is not specific to the subgoal reward model in Equation (9.1) but is really a consequence of the softmax transformation in Equation (9.2). In fact, the same problem occurs when the model is applied in a regular MDP environment with arbitrary reward function, for example, when the agent is provided an additional constant reward at all states. Clearly, such a constant reward provides no further information about the underlying task and should hence not affect the agent's belief about the optimal choice of actions (compare discussion on constant reward functions and transformations of rewards in [NR00; NHR99]).

Based on these two observations, our intuition tells us that we seek for a rationality model that is *invariant to affine transformations of the reward signal*, meaning that any two reward functions $R : \mathcal{S} \rightarrow \mathbb{R}$ and $\bar{R} \triangleq xR + y$ with $x \in (0, \infty)$, $y \in \mathbb{R}$, should give rise to the same intentional representation. As we shall see in Section 10.1.3, this can be achieved by modeling the behavior of an agent based on the *relative advantages* of actions rather than on their absolute expected returns.

10.1.2 Impact of the Transition Dynamics

The second implication of the softmax likelihood model is less immediate and inherently tied to the dynamics of the system. To explain the problem, we consider a scenario where we have a precise idea about the potential goals of the expert. For our example, we adopt the Gridworld dynamics described in Section 11.1 and consider a simple upward-directed trajectory of state-action pairs, which we aspire to explain using a single (sub-)goal. The complete setting is depicted in Figure 10.1.

Intuitively, the shown demonstration set should lead to goals that are located in the upper region of the state space and concentrated around the vertical center line. Moreover, as we move away from that center line, we expect to observe a smooth decrease in the subgoal likelihood, while the rate of the decay should reflect our assumed level of confidence in the expert. As it turns out, the induced BNIRL subgoal posterior distribution, shown in the top row for different values of β , contradicts this intuition. In particular, we observe that the model yields unreasonably high posterior values at the upper border states and corners of the state space, which, according to our intuitive understanding of the problem, cannot be justified by the given demonstration set.

To pin down the cause of this effect, we recall from Equation (9.2) that the likelihood of an action grows with the corresponding Q-value. Hence, we need to ask what causes the Q-values of the demonstrated actions to be large when the subgoal is assumed to be located at one of the upper corner/border states of the space. Using Bellman's principle, we can express the optimal Q-function for any subgoal g as

$$\begin{aligned} Q^*(s, a | g) &= R_g(s) + \gamma \mathbb{E}_{\mathcal{T}}[V^*(s' | g) | s, a] \\ &= R_g(s) + \gamma \mathbb{E}_{\mathcal{T}}[\mathbb{E}_{\rho^{\pi_g}}[R_g(s'') | s'] | s, a] \\ &= R_g(s) + \gamma \mathbb{E}_{\mathcal{T}}[C \rho^{\pi_g}(g | s') | s, a], \end{aligned} \tag{10.1}$$

where $V^*(s | g) \triangleq \max_{a \in \mathcal{A}} Q^*(s, a | g)$, $\pi_g(s) \triangleq \arg \max_{a \in \mathcal{A}} Q^*(s, a | g)$ is the optimal policy for subgoal g , and C is the subgoal reward from Equation (9.1). Lastly, $\rho^{\pi_g}(s' | s) \triangleq \sum_{t=0}^{\infty} \gamma^t p_t(s' | s, \pi_g)$ denotes the (improper) discounted state distribution generated by executing policy π_g from the considered initial state s , where $p_t(s' | s, \pi_g)$ refers to the probability of reaching state s' from state s under policy π_g after exactly t steps, which is defined implicitly via the transition model \mathcal{T} .

The outer expectation in Equation (10.1) accounts for the stochastic transition to the successor state s' , while the inner expectation evaluates the expected cumulative reward

over all states s'' that are reachable from s' . It is important to note that — by the construction of the Q-function — only the first move of the agent to state s' depends on the choice of action a whereas all remaining moves (i.e., the argument of the expectation in the last line) are purely determined by the system dynamics and the subgoal policy π_g . Focusing on that inner part, we conclude that, *regardless of the chosen action a* , the Q-values will be large whenever the assumed subgoal induces a high state visitation frequency ρ^{π_g} at its own location g . The latter is fulfilled if

- (i) the chance of reaching the goal in a small number of steps is high so that the effect of discounting is small and/or
- (ii) the controlled transition dynamics $\mathcal{T}(s' | s, \pi_g(s))$ that are induced by the subgoal lead to a high chance of hitting the goal frequently.

Note that the first condition implies that the model generally prefers subgoals that are close to the demonstration set — a property that cannot be justified in all cases. For example, the recording of the demonstrations could have simply ended before the expert was able to reach the goal (see Figure 9.2). Yet, if desired, this proximity property should be more naturally attributed to the subgoal prior model $p_g(g | \mathbf{s})$.

Moreover, we observe that the second condition depends primarily on the system dynamics \mathcal{T} , which can be more or less strongly influenced by the actions of the agent, depending on the scenario. In fact, in a pathological example, \mathcal{T} could be even independent of the agent’s decisions, meaning that the agent has no control over its state. An example illustrating this extreme case would be a scenario where the agent gets always driven to the same terminal state, regardless of the executed policy. Although it is somewhat pointless speak of “subgoals” in this context, that terminal state would exhibit a high subgoal likelihood according to the softmax model because the corresponding visitation frequency would be inevitably large. A softened variant of this condition can occur at corner/border states (i.e., states in which the agent experiences fewer degrees of freedom and which are hence more difficult to leave than others) and transition states (i.e., states that must be passed in order to get from certain regions of the space to others), which naturally exhibit an increased visitation frequency due to the characteristics of the environment.

In our example in Figure 10.1, we can observe the symptoms of both described conditions clearly. In particular, for an upward-directed policy as it is implied by the shown demonstration set, the induced state visitation distribution exhibits increased values at exactly the aforementioned border and corner states (due to the reflections occurring

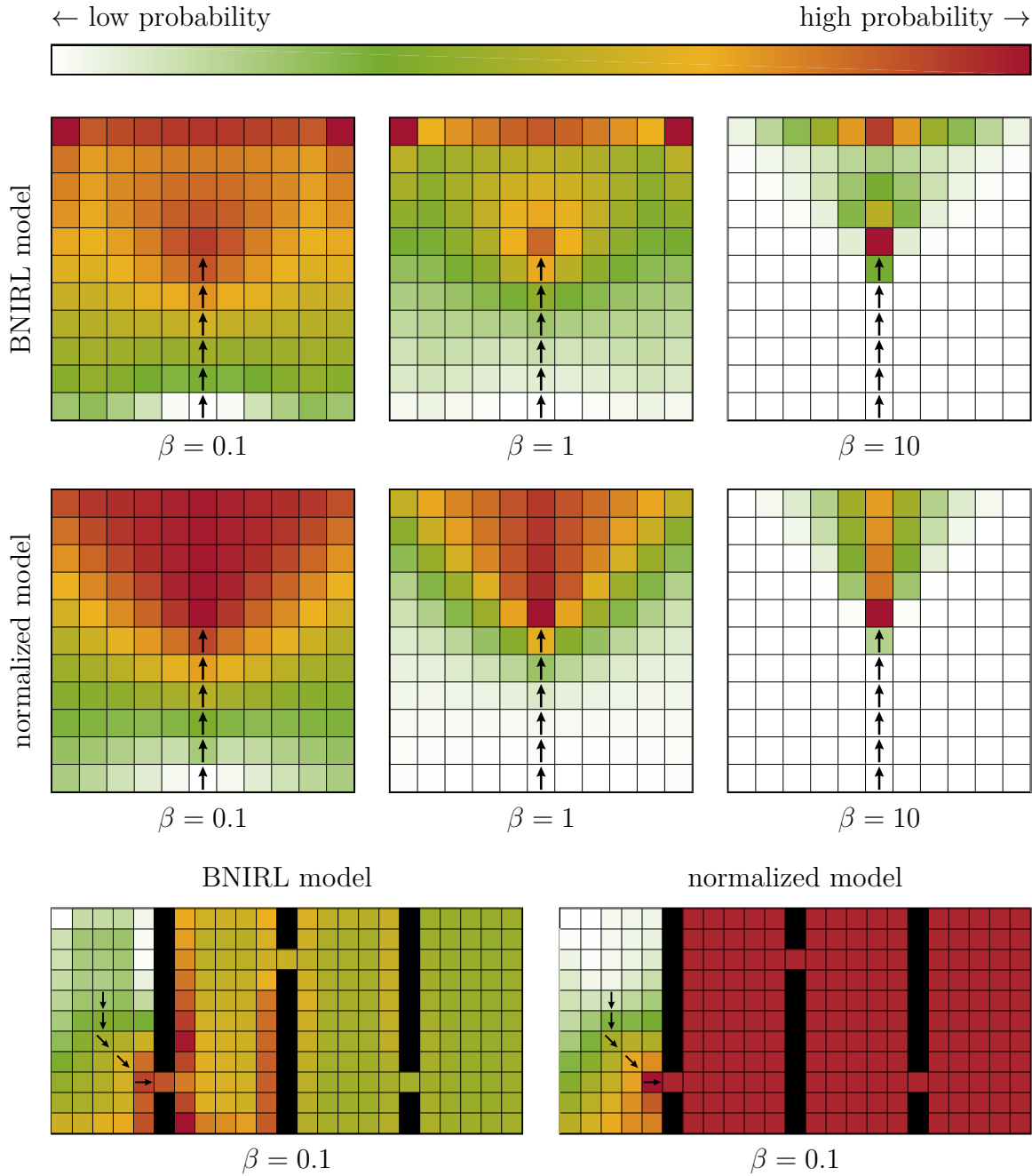


Figure 10.1: Comparison of the subgoal posterior distributions induced by the original BNIRL likelihood model and by the proposed normalized model, based on the Gridworld dynamics described in Section 11.1 and a uniform subgoal prior distribution p_g . The range of the shown color scheme is to be understood per subfigure. Black squares indicate wall states. The BNIRL likelihood model yields unreasonably high subgoal posterior mass at the border states and corners of the state space (due to locally increased state visitation probabilities arising from wall reflections) as well as at trajectory endings (caused by the implicit proximity property of the model) — see Section 10.1.2 for details. Both effects are mitigated by the proposed normalized likelihood model, which describes the action-selection process of the agent using relative advantages of actions instead of absolute returns.

to the agent when hitting the state space boundary) as well as close to the trajectory ending (caused by the proximity condition).

10.1.3 The Normalized Likelihood Model

To address these problems, we modify the likelihood model using a rescaling of the involved Q-values. Let $Q^\wedge(s|g)$ and $Q^\vee(s|g)$ denote the maximum and minimum Q-values at state s for subgoal g , i.e., $Q^\wedge(s|g) \triangleq \max_{a \in \mathcal{A}} Q^*(s, a|g)$ and $Q^\vee(s|g) \triangleq \min_{a \in \mathcal{A}} Q^*(s, a|g)$. We then define the normalized state-action value function $Q^\bullet : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ as

$$Q^\bullet(s, a|g) \triangleq \begin{cases} \frac{Q^*(s, a|g) - Q^\vee(s|g)}{Q^\wedge(s|g) - Q^\vee(s|g)} & \text{if } Q^\wedge(s|g) \neq Q^\vee(s|g), \\ \epsilon & \text{otherwise,} \end{cases} \quad (10.2)$$

where $\epsilon \in (0, 1]$ is an arbitrary constant that is canceled out in Equation (10.3). In contrast to the Bellman state-action value function Q^* , which quantifies the expected return of an action, the normalized function Q^\bullet assesses the return of that action in relation to the returns of all other actions. This concept is similar to that of the advantage function [Bai93] with the important difference that the values returned by Q^\bullet are normalized to the range $[0, 1]$ and thus serve as an indicator for the *relative* quality of actions. Accordingly, the values can be interpreted as *relative advantages* (i.e., relative to the maximum possible advantage among all actions). The normalized subgoal likelihood model is then constructed analogously to the BNIRL likelihood model, i.e.

$$\pi^\bullet(a_d | s_d, g_{\bar{z}_d}) \propto \exp \left\{ \beta Q^\bullet(s_d, a_d | g_{\bar{z}_d}) \right\}. \quad (10.3)$$

The key property of this model is that it is invariant to affine transformations of the reward function, as summarized by the following proposition.

Proposition 1 (Affine invariance). *Consider an MDP with reward function $R : \mathcal{S} \rightarrow \mathbb{R}$ and let $Q^*(s, a | R)$ denote the corresponding optimal state-action value function. For the corresponding normalized function Q^\bullet it holds that $Q^\bullet(s, a | R) = Q^\bullet(s, a | xR + y) \forall x \in (0, \infty), y \in \mathbb{R}, s \in \mathcal{S}, a \in \mathcal{A}$. Hence, the subgoal likelihood model in Equation (10.3) is invariant to affine transformations of R .*

Proof. Due to the linear dependence of Q^* on the reward function R (Equation 9.3) it holds that $Q^*(s, a | xR + y) = xQ^*(s, a | R) + \frac{y}{1-\gamma}$. Using this relationship in Equation (10.2), it follows immediately that $Q^\bullet(s, a | R) = Q^\bullet(s, a | xR + y)$. \square

Using the proposed likelihood model offers several advantages. First of all, it enables a more generic choice of the uncertainty coefficient β (Section 10.1.1). This is because the returned Q^\bullet -values lie in the fixed range $[0,1]$, where 0 always indicates the lowest and 1 indicates the highest confidence. For example, setting $\beta = \log(\beta')$ for some $\beta' \in (0, \infty)$ always corresponds to the assumption that the expert chooses the optimal action with a probability that is β' times higher than the probability of choosing the least favorable action, irrespective of the underlying system model.

Moreover, as the results in Figure 10.1 reveal, the induced subgoal posterior distribution is notably closer to our expectation. The reason for this is twofold: first, a likelihood computation based on relative advantages mitigates the influence of the transition dynamics discussed in Section 10.1.2. This is because the described cumulation effect of the state visitation distribution ρ^{π_g} (Equation 10.1) is present in the returns of all actions and is thus reduced through the proposed normalization. For instance, if the agent in our Gridworld follows a policy that is all upward directed (as shown in the example), the induced state visitation distribution exhibits increased values at the upper border states of the world, even if we manipulated the first action of the agent (as considered in the Bellman Q -function). Accordingly, the original model would indicate an increased subgoal likelihood at those states. The normalized model, by contrast, is less affected as it constructs the likelihood by considering the increased visitation frequencies *relative* to each other.

Second, since the normalization diminishes the effect of the discounting, the subgoal posterior distribution is less concentrated around the trajectory ending and shows significant mass along the extrapolated path of the agent. This property allows us to identify far located states as potential goal locations, which adds more flexibility to the inferred subgoal constellation (compare Figure 9.2). As an illustrating example, consider the scenario shown in the bottom part of Figure 10.1. We observe that the normalized model assigns high posterior mass to all states in the right three corridors since any subgoal located in those corridors explains the demonstration set equally well. Here, the difference between the two models is even more pronounced because the transition dynamics have a strong impact on the agent behavior due to the added wall states. For further details, we refer to Section 11.1, where we provide additional insights into the subgoal inference mechanism.

10.2 Modeling Time-Invariant Intentions

With our redesigned likelihood model, we now focus on the partitioning structure of the model. Herein, we first consider the case where the intentions of the agent are constant with respect to time. As explained in Limitation 3, this setting is consistent with the standard MDP formalism (Chapter 2) in the sense that the optimal policy for the considered task can be described in the form of a state-to-action mapping.

As a first step, to account for this relation, we establish a link between the model partitioning structure and the underlying system state space by replacing the demonstration-based indicators $\tilde{\mathbf{z}} = \{\tilde{z}_d \in \mathbb{N}\}_{d=1}^D$ with a new set of variables $\mathbf{z} \triangleq \{z_i \in \mathbb{N}\}_{i=1}^{|\mathcal{S}|}$. Unlike $\tilde{\mathbf{z}}$, these new indicators do not operate directly on the data but are instead tied to the elements in \mathcal{S} . Although they formally represent a new type of variable, we can still imagine that their distribution follows a CRP. This yields an intermediate model of the form

$$p(\mathbf{a}, \mathbf{z}, \mathcal{G} | \mathbf{s}) = p(\mathbf{z}) \prod_{k=1}^{\infty} p_g(g_k | \mathbf{s}) \prod_{d=1}^D \pi^\bullet(a_d | s_d, g_{\mathbf{z}_{s_d}}),$$

whose structure is illustrated in Figure 10.2b. To see the difference to Equation (9.4), notice the way the subgoals are indexed in this model. The model can be considered as the intentional counterpart of the subintentional model in Equation (4.11) if we neglect the fact that the present formulation is based on state-action demonstrations instead of state trajectories (see notation hint on page 71).

The intermediate model makes it possible to reason about the policy (or, more suggestively, the underlying state-to-action rule approximated by the expert) at visited parts of the state space. Yet, the model is unable to extrapolate the gathered information to unvisited states, for the reasons explained in Section 9.2. This problem can be solved by replacing the exchangeable prior distribution over subgoal assignments induced by the CRP with a non-exchangeable one, in order to account explicitly for the covariate state information contained in the demonstration set. Based on our insights from Part I, we use the ddCRP for this purpose, which allows a very intuitive handling of the state context. For alternatives, we point to the survey paper by Foti and Williamson [FW15] (see *Discussion and Outlook*).

As explained in Section 5.2, the clustering mechanism in this model is described via stochastic state-to-state assignments, represented by a set of indicator variables $\mathbf{c} = \{c_i \in \mathcal{S}\}_{i=1}^{|\mathcal{S}|}$ whose joint distribution factors into a set of marginals defined as in Equation (5.3). Note that the distances $\{\Delta_{i,j}\}$ can be obtained via a suitable metric

defined on the state space, which may be furthermore used for calibrating the score function f (see subsequent section). The state partitioning structure itself is then determined by the connected components of the induced ddCRP graph (Figure 5.1). Our joint distribution, visualized in Figure 10.2c, thus reads as

$$p(\mathbf{a}, \mathbf{c}, \mathcal{G} | \mathbf{s}) = p(\mathbf{c}) \prod_{k=1}^{\infty} p_g(g_k | \mathbf{s}) \prod_{d=1}^D \pi^{\bullet}(a_d | s_d, g_{\mathbf{z}(\mathbf{c})|_{s_d}}), \quad (10.4)$$

where $\mathbf{z}(\mathbf{c})|_s$ denotes the subgoal label of state s arising from the considered indicator set \mathbf{c} . In order to highlight the state dependence of the underlying subgoal mechanism, we refer to this model as ddBNIRL-S.

10.2.1 The Canonical State Metric for Spatial Subgoal Modeling

The use of the ddCRP as a prior model for the state partitioning in Equation (10.4) inevitably requires some notion of distance between any two states of the system, in order to compute the involved function scores $\{f(\Delta_{i,j})\}$. When no such distances are provided by the problem setting (see Limitation 2, second point), a suitable (quasi-)metric can be derived from the transition dynamics of the system, which turns out to be the canonical choice for the ddBNIRL-S model.

Consider the Markov chain governing the state process $\{s_{t=n}\}_{n=1}^{\infty}$ of an agent for some specific policy π . For any ordered pair of states (i, j) , the chain naturally induces a value $T_{i \rightarrow j}^{\pi}$, called a *hitting time* [TK14; TB08], which represents the expected number of steps required until the state process, initialized at i , eventually reaches state j for the first time, i.e.

$$T_{i \rightarrow j}^{\pi} \triangleq \mathbb{E} \left[\min \{n \in \mathbb{N} : s_{t=n} = j\} \mid s_0 = i, \pi \right].$$

In the context of our subgoal problem, the natural quasi-metric to measure the directed distance between two states i and j is thus given by the time it takes to reach the goal state j from the starting state i under the corresponding optimal subgoal policy $\pi_j(s) = \arg \max_{a \in \mathcal{A}} Q^*(s, a | j)$, i.e., $\Delta_{i,j} \triangleq T_{i \rightarrow j}^{\pi_j}$.

For ddBNIRL-S (as well as for the waypoint method in BNIRL), this choice is particularly appealing since the subgoal policies $\{\pi_j\}$ are already available within the inference procedure after the state-action values have been computed for the likelihood model (more on this in Section 10.4.5). The corresponding distances $\{\Delta_{i,j}\}$ can be obtained efficiently in a single policy evaluation step since $\Delta_{i,j}$ corresponds to the optimal

(negative) expected return at the starting state i for the special setting where the respective target state j is made absorbing with zero reward while all other states are assigned a reward of -1 .

10.2.2 Choice of the Score Function

From Equation (5.3) it is evident that the ddCRP model favors partitioning structures that result from the connection of nearby states. In the context of the subgoal problem, this property translates to the prior assumption that, most likely, each subgoal is approached by the expert from only one specific localized region in the system state space. While this assumption may be reasonable for some tasks, other tasks require that certain target states be approached more than one time, from different regions in the system state space. In such cases, it is beneficial if the model can reuse the same subgoal in various contexts, in order to obtain a more efficient task encoding (Figure 9.2).

From a mathematical point of view, the prerequisite for learning such encodings is that the score function f does not shrink to zero at large distance values, so that there remains a non-zero probability of connecting states that are far apart from each other. This can be achieved, for example, by representing f as a convex combination of a monotone decreasing zero-approaching function $\bar{f} : [0, \infty) \rightarrow [0, \infty)$ and some constant offset $\kappa \in (0, 1]$, i.e.

$$f(\Delta) = (1 - \kappa)\bar{f}(\Delta) + \kappa,$$

where \bar{f} is chosen, e.g., as a radial basis function (Equation 6.3). Note that, in order to implement a desired degree of locality in the model, the scale of the decay function f (or \bar{f} , respectively) can be further calibrated based on the quantiles of the distribution of the given distances $\{\Delta_{i,j}\}$.

10.3 Modeling Time-Varying Intentions

For the case of changing expert intentions, we need to keep the flexibility of BNIRL to select a new subgoal at each decision instant, instead of restricting our policy to target a unique subgoal per state (Figure 9.3). Hence, we retain the basic BNIRL structure in this case and define the subgoal allocation mechanism using a set of data-related indicator variables. However, in contrast to BNIRL, which makes no assumptions about the temporal relationship of the subgoals and thus allows arbitrary changes of the expert's intentions (Limitation 2), we design our joint distribution in a way that favors

smooth action plans in which the expert persistently follows a subgoal over an extended period of time.

Again, we can make use of the ddCRP properties to encode the underlying smoothness assumption, but this time using a score function defined on the *temporal distance* between demonstration pairs. For this purpose, we require an additional piece of information, namely the unique timestamp of each demonstration example. Accordingly, we need to assume that our data set is of the form $\tilde{\mathcal{D}} \triangleq \{(s_d, a_d, t_d)\}_{d=1}^D$, where t_d denotes the recording time of the d th demonstration pair (s_d, a_d) .^{*}

The prior distribution over data partitionings can then be written as $p(\tilde{\mathbf{c}}) = \prod_{d=1}^D p(\tilde{c}_d)$,

$$p(\tilde{c}_d = d') \propto \begin{cases} \nu & \text{if } d = d', \\ f(\tilde{\Delta}_{d,d'}) & \text{otherwise,} \end{cases}$$

where the indices $d, d' \in \{1, \dots, D\}$ range over the size of the demonstration set. Herein, $\tilde{\Delta}_{d,d'} \triangleq |t_d - t_{d'}|$ denotes the temporal distance between the data points d and d' . As before, we use the “ \sim ”-notation to distinguish the data-related partitioning variables $\tilde{\mathbf{c}}$, $\tilde{\mathbf{z}}$ and distances $\{\tilde{\Delta}_{i,j}\}$ from their state-space-related counterparts \mathbf{c} , \mathbf{z} and $\{\Delta_{d,d'}\}$ used in ddBNIRL-S. Note, however, that the score function f is independent of the underlying model type and may be chosen as described in Section 10.2.1, with a scale calibrated to the duration of the demonstrated task. With that, we obtain our temporal subgoal model as

$$p(\mathbf{a}, \tilde{\mathbf{c}}, \mathcal{G} \mid \mathbf{s}) = p(\tilde{\mathbf{c}}) \prod_{k=1}^{\infty} p_g(g_k \mid \mathbf{s}) \prod_{d=1}^D \pi^{\bullet}(a_d \mid s_d, g_{\tilde{\mathbf{z}}(\tilde{\mathbf{c}})|_d}), \quad (10.5)$$

where $\tilde{\mathbf{z}}(\tilde{\mathbf{c}})|_d$ refers to the subgoal label of the d th demonstration pair induced by the given assignment $\tilde{\mathbf{c}}$. Analogous to our spatial subgoal model, we refer to this model as ddBNIRL-T. The structural differences between all models can be seen from Figure 10.2.

10.3.1 Relationship to BNIRL

Since the distance-dependent CRP contains the classical CRP as a special case for a specific choice of distance metric and score function [BF11], the ddBNIRL-T model can

^{*} Note that the timestamps $\{t_d\}$ are naturally available if the demonstrations are recorded in trajectory form, where we observe several consecutive state-action pairs. In fact, the temporal information of the data is also required for the waypoint method to work (Limitation 2), even though the authors of BNIRL formally assume to have access to the reduced data set of state-action pairs only.

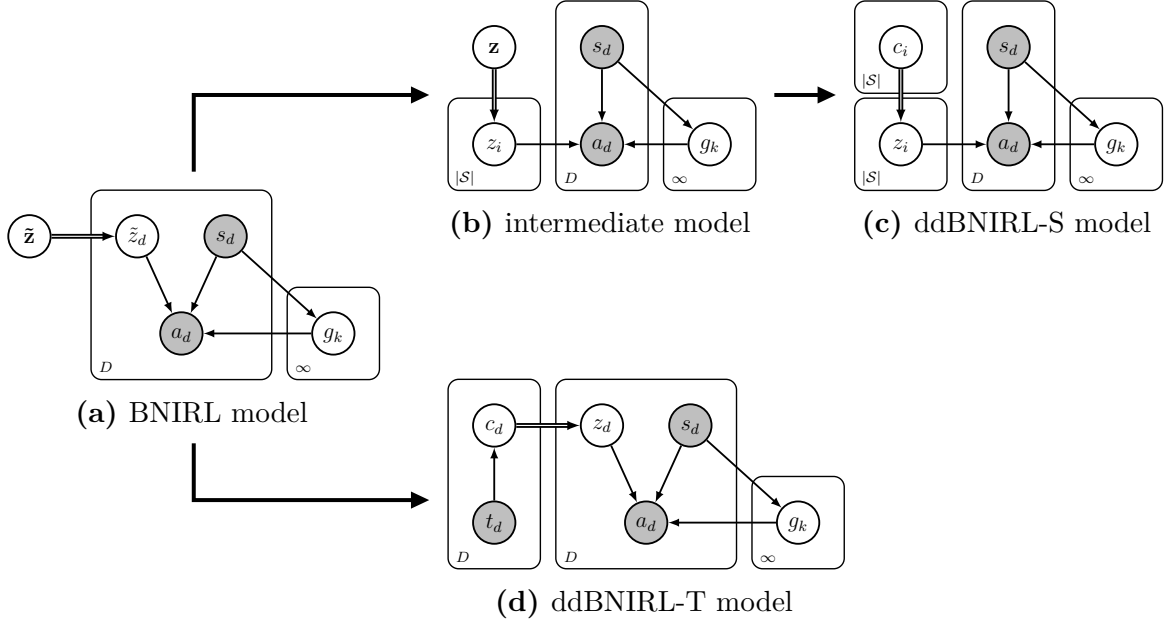


Figure 10.2: Relationships between all discussed subgoal models, illustrated in the form of Bayesian networks. Shaded nodes represent observed variables; deterministic dependencies are highlighted using double strokes.

be considered a strict generalization of the original BNIRL framework (neglecting the likelihood normalization in Section 10.1). In the same way, ddbNIRL-S generalizes the intermediate model presented in Section 10.2 (see Figure 10.2). However, although the BNIRL model can be recovered from ddbNIRL, it is important to note that the sampling mechanisms of both frameworks are fundamentally different. Whereas in BNIRL the subgoal assignments are sampled directly, the clustering structure in ddbNIRL is defined *implicitly* via the assignment variables \mathbf{c} and $\tilde{\mathbf{c}}$, respectively. As explained by Blei and Frazier [BF11], this has the effect that the Markov chain governing the Gibbs sampler mixes significantly faster because several cluster assignments can be altered in a single step, which effectively realizes a blocked Gibbs sampler [RS97].

10.4 Prediction and Inference

Having introduced the ddbNIRL framework, we now explain how it can be used to generalize a given expert behavior. To this end, we first focus on the task of action prediction at a given query state, and then explain in a second step how to extract the necessary information from the demonstration data. Along the way, we also give insights into the implicit intentional model learned through the framework.

Note: In order to keep the level of redundancy at a minimum, the following considerations are based on the ddBNIRL-S model (Section 10.2). The results for ddBNIRL-T (Section 10.3) follow straightforwardly; the only change in the equations is the way the subgoals are referenced. To obtain the corresponding expressions, we simply replace the assignment variables \mathbf{c} with $\tilde{\mathbf{c}}$ and change the cluster definition in Equation (10.10) to $\mathcal{C}_k \triangleq \{d \in \{1, \dots, D\} : \tilde{\mathbf{z}}(\tilde{\mathbf{c}})|_d = k\}$. Accordingly, all occurrences of $\mathbf{z}(\mathbf{c})|_{s^*}$ change to $\tilde{\mathbf{z}}(\tilde{\mathbf{c}})|_{d^*}$, $\mathbf{z}(\mathbf{c})|_{s_d}$ becomes $\tilde{\mathbf{z}}(\tilde{\mathbf{c}})|_d$, and $s_d \in \mathcal{C}_k$ is replaced with $d \in \mathcal{C}_k$.

10.4.1 Action Prediction

Similar to the work by Abbeel and Ng [AN04], we consider the task of predicting an action $a^* \in \mathcal{A}$ at some query state $s^* \in \mathcal{S}$ that is optimal with respect to the expert’s *unknown* reward model. However, in contrast to most existing IRL methods, our approach is not based on point estimates of the expert’s reward function but takes into account the entire hypothesis space of reward models. This allows us to obtain the full posterior predictive policy from the expert data.

Mathematically, the task is formulated as computing the predictive action distribution $p(a^* | s^*, \mathcal{D})$, which captures the full information about the expert behavior contained in the demonstration set \mathcal{D} . We start by expanding that distribution with the help of the latent state assignments \mathbf{c} , i.e.

$$p(a^* | s^*, \mathcal{D}) = \sum_{\mathbf{c} \in \mathcal{S}^{|\mathcal{S}|}} p(a^* | s^*, \mathcal{D}, \mathbf{c}) p(\mathbf{c} | \mathcal{D}).$$

The conditional distribution $p(a^* | s^*, \mathcal{D}, \mathbf{c})$ can be expressed in terms of the posterior distribution of the subgoal targeted at the query state s^* , i.e.

$$p(a^* | s^*, \mathcal{D}) = \sum_{\mathbf{c} \in \mathcal{S}^{|\mathcal{S}|}} p(\mathbf{c} | \mathcal{D}) \sum_{i \in \mathcal{S}} p(a^* | s^*, \mathbf{c}, g_{\mathbf{z}(\mathbf{c})|_{s^*}} = i) p(g_{\mathbf{z}(\mathbf{c})|_{s^*}} = i | \mathcal{D}, \mathbf{c}),$$

where we used the fact that the prediction a^* is conditionally independent of the demonstration set \mathcal{D} given the state partitioning structure and the corresponding subgoal assigned to s^* (that is, given \mathbf{c} and $g_{\mathbf{z}(\mathbf{c})|_{s^*}}$).

From the joint distribution in Equation (10.4), it follows that

$$p(g_k | \mathcal{D}, \mathbf{c}) = \frac{1}{Z_k(\mathcal{D}, \mathbf{c})} p_g(g_k | \mathbf{s}) \prod_{d: \mathbf{z}(\mathbf{c})|_{s_d} = k} \pi(a_d | s_d, g_k), \quad (10.6)$$

where $Z_k(\mathcal{D}, \mathbf{c})$ is the corresponding normalizing constant, i.e.

$$Z_k(\mathcal{D}, \mathbf{c}) \triangleq \sum_{i \in \text{supp}(p_g)} p_g(g_k = i | \mathbf{s}) \prod_{d: \mathbf{z}(\mathbf{c})|_{s_d} = k} \pi(a_d | s_d, g_k = i). \quad (10.7)$$

Using this relationship, we get

$$\begin{aligned} p(a^* | s^*, \mathcal{D}) &= \sum_{\mathbf{c} \in \mathcal{S}^{|\mathcal{S}|}} \frac{1}{Z_k(\mathcal{D}, \mathbf{c})} p(\mathbf{c} | \mathcal{D}) \sum_{i \in \text{supp}(p_g)} p_g(g_{\mathbf{z}(\mathbf{c})|_{s^*}} = i | \mathbf{s}) \dots \\ &\dots \times \prod_{d: \mathbf{z}(\mathbf{c})|_{s_d} = \mathbf{z}(\mathbf{c})|_{s^*}} \pi(a_d | s_d, g_{\mathbf{z}(\mathbf{c})|_{s^*}} = i) p(a^* | s^*, \mathbf{c}, g_{\mathbf{z}(\mathbf{c})|_{s^*}} = i). \end{aligned}$$

In contrast to the summation over subgoal locations i , whose computational complexity is determined by the support of the subgoal prior distribution p_g and which grows at most linearly with the size of \mathcal{S} , the marginalization with respect to the indicator variables \mathbf{c} involves the summation of $|\mathcal{S}|^{|\mathcal{S}|}$ terms and becomes quickly intractable even for small state spaces. Therefore, we approximate this operation via Monte Carlo integration, which yields

$$p(a^* | s^*, \mathcal{D}) \approx \frac{1}{N} \sum_{n=1}^N \sum_{i \in \text{supp}(p_g)} p(g_{\mathbf{z}(\mathbf{c}^{\{n\}})|_{s^*}} = i | \mathcal{D}, \mathbf{c}^{\{n\}}) p(a^* | s^*, \mathbf{c}^{\{n\}}, g_{\mathbf{z}(\mathbf{c}^{\{n\}})|_{s^*}} = i),$$

where $\mathbf{c}^{\{n\}} \sim p(\mathbf{c} | \mathcal{D})$. The final prediction step can then be performed, for example, via the MAP policy estimate, i.e.

$$\hat{\pi}(s^*) \triangleq \arg \max_{a^* \in \mathcal{A}} p(a^* | s^*, \mathcal{D}). \quad (10.8)$$

The inference task, hence, reduces to the computation of the posterior samples $\{\mathbf{c}^{\{n\}}\}$, which is described in the next section.

10.4.2 Partition Inference

Based on the joint model in Equation (10.4), we obtain the posterior distribution $p(\mathbf{c} | \mathcal{D})$ in factorized form as

$$\begin{aligned} p(\mathbf{c} | \mathcal{D}) &= p(\mathbf{c}) \prod_{k=1}^{\infty} \sum_{g_k \in \text{supp}(p_g)} p_g(g_k | \mathbf{s}) \prod_{d=1}^D \pi(a_d | s_d, g_{\mathbf{z}(\mathbf{c})|_{s_d}}) \\ &= p(\mathbf{c}) \prod_{k=1}^{|\mathbf{z}(\mathbf{c})|} \sum_{g_k \in \text{supp}(p_g)} p_g(g_k | \mathbf{s}) \prod_{d: s_d \in \mathcal{C}_k} \pi(a_d | s_d, g_k), \end{aligned} \quad (10.9)$$

where \mathcal{C}_k denotes the k th state cluster induced by the assignment \mathbf{c} , i.e.

$$\mathcal{C}_k \triangleq \{s \in \mathcal{S} : \mathbf{z}(\mathbf{c})|_s = k\}, \quad (10.10)$$

and $|\mathbf{z}(\mathbf{c})|$ is the total number of clusters defined by \mathbf{c} . As explained by Blei and Frazier [BF11], the indicator samples $\{\mathbf{c}^{\{n\}}\}$ can be efficiently generated using a fast-mixing Gibbs chain. Starting from a given ddCRP graph defined by the subset of indicators $\mathbf{c}_{\setminus i} \triangleq \{c_j\} \setminus c_i$, the insertion of an additional edge c_i will result in one of three possible outcomes, as illustrated in Figure 5.1: in the case of adding a self-loop ($c_i = i$), the underlying partitioning structure stays unaffected. Setting $c_i \neq i$ either leaves the structure unchanged (if the target state is already in the same cluster as state i) or creates a new link between two clusters. In the latter case, the involved clusters are merged, which corresponds to a merging of the associated sums in Equation (10.9).

According to these three cases, the conditional distribution for the Gibbs procedure is obtained as

$$p(c_i = j \mid \mathbf{c}_{\setminus i}, \mathcal{D}) \propto \begin{cases} \nu & \text{if } i = j, \\ f(d_{i,j}) & \text{if no clusters are merged,} \\ f(d_{i,j}) \frac{\mathcal{L}(\mathcal{C}_{z_i} \cup \mathcal{C}_{z_j})}{\mathcal{L}(\mathcal{C}_{z_i}) \cdot \mathcal{L}(\mathcal{C}_{z_j})} & \text{if clusters } \mathcal{C}_{z_i} \text{ and } \mathcal{C}_{z_j} \text{ are merged.} \end{cases} \quad (10.11)$$

Herein, $\mathcal{L}(\mathcal{C})$ denotes the marginal action likelihood of all demonstrations accumulated in cluster \mathcal{C} , i.e.

$$\mathcal{L}(\mathcal{C}) = \sum_{g \in \text{supp}(p_g)} p_g(g \mid \mathbf{s}) \prod_{d: s_d \in \mathcal{C}} \pi(a_d \mid s_d, g), \quad (10.12)$$

which further represents the normalizing constant for the posterior distribution of the cluster subgoal (Equation 10.7). Accordingly, the fraction in Equation (10.11) can be interpreted as the likelihood ratio of the partitioning defined by $\mathbf{c}_{\setminus i}$ and the merged structure after inserting the new edge c_i .

10.4.3 Subgoal Inference

It is important to note that the inference method described in Sections 10.4.1 and 10.4.2 is based on a collapsed sampling scheme where all subgoals of our model are marginalized out. In fact, the ddBNIRL framework differs from BNIRL and other IRL methods in that the reward model of the expert is never made explicit for predicting new actions. Nonetheless, if desired (e.g., for the purpose of analyzing the expert's intentions), an estimate of the subgoal locations can be obtained in a post-hoc fashion from the subgoal

posterior distribution in Equation (10.6) for any given assignment \mathbf{c} . Examples are provided in Figure 11.1.

10.4.4 Action Inference

As mentioned in Limitation 5, the original BNIRL algorithm requires complete knowledge of the expert’s action record \mathbf{a} , which limits the range of potential application scenarios. For this reason, we generalize our inference scheme to the case where we have access to state information only, provided in the form of an alternative data set $\overline{\mathcal{D}} \triangleq \{(s_d, \bar{s}_d)\}_{d=1}^D$, where \bar{s}_d refers to the state visited by the expert immediately after s_d . In this setting, inference can be performed by extending the Gibbs procedure with an additional collapsed sampling stage, i.e.

$$p(a_d | \mathbf{a}_{\setminus d}, \overline{\mathcal{D}}, \mathbf{c}) \propto \mathcal{T}(\bar{s}_d | s_d, a_d) \sum_{i \in \text{supp}(p_g)} p_g(g_{\mathbf{z}(\mathbf{c})|s_d} = i) \prod_{d': \mathbf{z}(\mathbf{c})|_{s_{d'}} = \mathbf{z}(\mathbf{c})|_{s_d}} \pi(a_{d'} | s_{d'}, g_{\mathbf{z}(\mathbf{c})|s_d} = i), \quad (10.13)$$

which, for a fixed assignment \mathbf{c} , recovers an estimate of the latent action set \mathbf{a} from the observed state transitions. Note that knowledge of the transition model \mathcal{T} is required for this step as it provides the necessary link between the expert’s actions and the observed successor states. The same extension is possible for the ddBNIRL-T model, provided that the transition timestamps $\{t_d\}$ are known (Section 10.3).

10.4.5 Computational Complexity

As a last point in this section, we would like to discuss the computational complexity of our approach. For this purpose, here a quick reminder on the used notation: we write $|\mathcal{S}|$ and $|\mathcal{A}|$ for the cardinalities of the state and action space, respectively, and use the letter D for the size of the demonstration set. Further, we write \mathcal{C}_k to refer to the k th state cluster (ddBNIRL-S) or data cluster (ddBNIRL-T). In the subsequent paragraphs, we additionally use the notation $N_D(\mathcal{C}_k)$ to access the number of demonstration data points associated with cluster \mathcal{C}_k , K to indicate the number of clusters in the current iteration, $N_g \triangleq |\text{supp}(p_g)|$ as a shorthand for the size of the support of the subgoal prior distribution, and N_c for the number of indicator variables, i.e., $N_c \triangleq |\mathcal{S}|$ for ddBNIRL-S and $N_c \triangleq D$ for ddBNIRL-T.

Initialization Phase

Common to all discussed models (including BNIRL) is that they depend on a preceding planning phase, where we compute, potentially in parallel, the state-action value functions (Equation 9.3) for all N_g considered subgoals, which allows us to construct the subgoal likelihood model (Equation 9.2 or Equation 10.3). The overall computational complexity of this procedure is of order $\mathcal{O}(N_g \mathbb{C}_{\text{MDP}}(|\mathcal{S}|, |\mathcal{A}|))$, where $\mathbb{C}_{\text{MDP}}(x, y)$ denotes the complexity of the used planning routine to (approximately) solve an MDP of size x with a total number of y actions. Using a value iteration algorithm, for instance, this can be achieved in $\mathcal{O}(\mathbb{C}_{\text{MDP}}(|\mathcal{S}|, |\mathcal{A}|)) = \mathcal{O}(|\mathcal{S}|^2 |\mathcal{A}|)$ steps [LDK95]. If we assume that the expert reaches all subgoals during the demonstration phase [MH12], we can restrict the support of the subgoal prior to the visited states, so that N_g is upper bounded by $\min(|\mathcal{S}|, D)$. Note that there exist approximation techniques that make the computation tractable in large/continuous state spaces (see *Discussion and Outlook*).

Before we start the sampling procedure, we compute all single-cluster likelihoods $\{\mathcal{L}(\mathcal{C}_k)\}$ and pairwise likelihoods $\{\mathcal{L}(\mathcal{C}_k \cup \mathcal{C}_{k'})\}$ according to Equation (10.12), based on some (random) initial cluster structure. The likelihood computation for the k th cluster \mathcal{C}_k involves a product over $N_D(\mathcal{C}_k)$ data points, which needs to be calculated for each of the N_g subgoals before taking their weighted average. This step has to be executed (potentially in parallel) for all clusters. However, because each demonstration is associated with exactly one cluster (either directly as in ddBNIRL-T or via the corresponding state variable as in ddBNIRL-S) and hence $\sum_k N_D(\mathcal{C}_k) = D$, the total complexity for computing all single-cluster likelihoods is of order $\mathcal{O}(N_g D)$, irrespective of the actual cluster structure. A similar line of reasoning applies to the computation of the pairwise likelihoods, yielding the same complexity order. Yet, for the latter we need to consider all possible cluster combinations. Assuming an initial number of K clusters, there are in total $K(K-1)/2$ pairwise likelihoods to be computed. Hence, the overall complexity of the initialization phase can be summarized as $\mathcal{O}(N_g D K^2)$.

Partition Inference

For the partition inference, the bulk of the computation lies in the repeated construction of the likelihood term in Equation (10.11), which needs to be updated whenever the cluster structure changes. To analyze the complexity, we consider the sampling step of an individual assignment variable c_i (or likewise \tilde{c}_i). In the worst case, removing the edge that belongs to c_i from the ddCRP graph divides the associated cluster into two parts (Figure 5.1), so that two new single-cluster likelihoods need to be computed. With the upper bound D on the number of data points associated with the cluster

before the division, this operation is of worst-case complexity $\mathcal{O}(N_g D)$ (see initialization phase). Irrespective of whether a division occurs, we then need to compute all pairwise cluster likelihoods with the (new) cluster connected via c_i . For a total of $K - 1$ possible choices, this is done in $\mathcal{O}(N_g DK)$ operations (see initialization phase).

After assigning the indicator, we move on to the next variable where the process repeats. If we assume, for simplicity, that the number of clusters stays constant during a full Gibbs cycle, the total complexity of updating all cluster assignments is hence of order $\mathcal{O}(N_g DK N_c)$. A (pessimistic) upper bound for the general case can be obtained by assuming that each data point defines its own cluster, in which case the complexity increases to $\mathcal{O}(N_g D^2 N_c)$. Note that, in order to identify the new cluster structure after changing an assignment, we additionally need to track the connected components of the underlying ddCRP graph. As explained by Kapron et al. [KKM13], this can be done in polylogarithmic worst-case time.

Action Sampling

In order to compute the conditional probability distribution of a particular action a_d , we need to evaluate a product involving all actions that belong to the same cluster as action a_d (Equation 10.13). First, we can compute the product over all actions except a_d itself, where the number of involved terms is again upper-bounded by D . Appending the term that belongs to a_d for all possible action choices requires another $|\mathcal{A}|$ operations. These two steps need to be repeated for all possible subgoals, yielding an upper bound on the complexity of order $\mathcal{O}(N_g(D + |\mathcal{A}|))$. For a full Gibbs cycle, which involves sampling all D action variables, the overall (worst-case) complexity is hence of order $\mathcal{O}(N_g(D + |\mathcal{A}|)D)$.

11

Experimental Results

In this section, we present experimental results for our framework. The evaluation is separated into four parts:

- (i) a proof of concept and conceptual comparison to BNIRL (Section 11.1),
- (ii) a performance comparison with related algorithms (Section 11.2),
- (iii) a real data experiment conducted on a KUKA robot (Section 11.3) and
- (iv) an active learning task (Section 11.4).

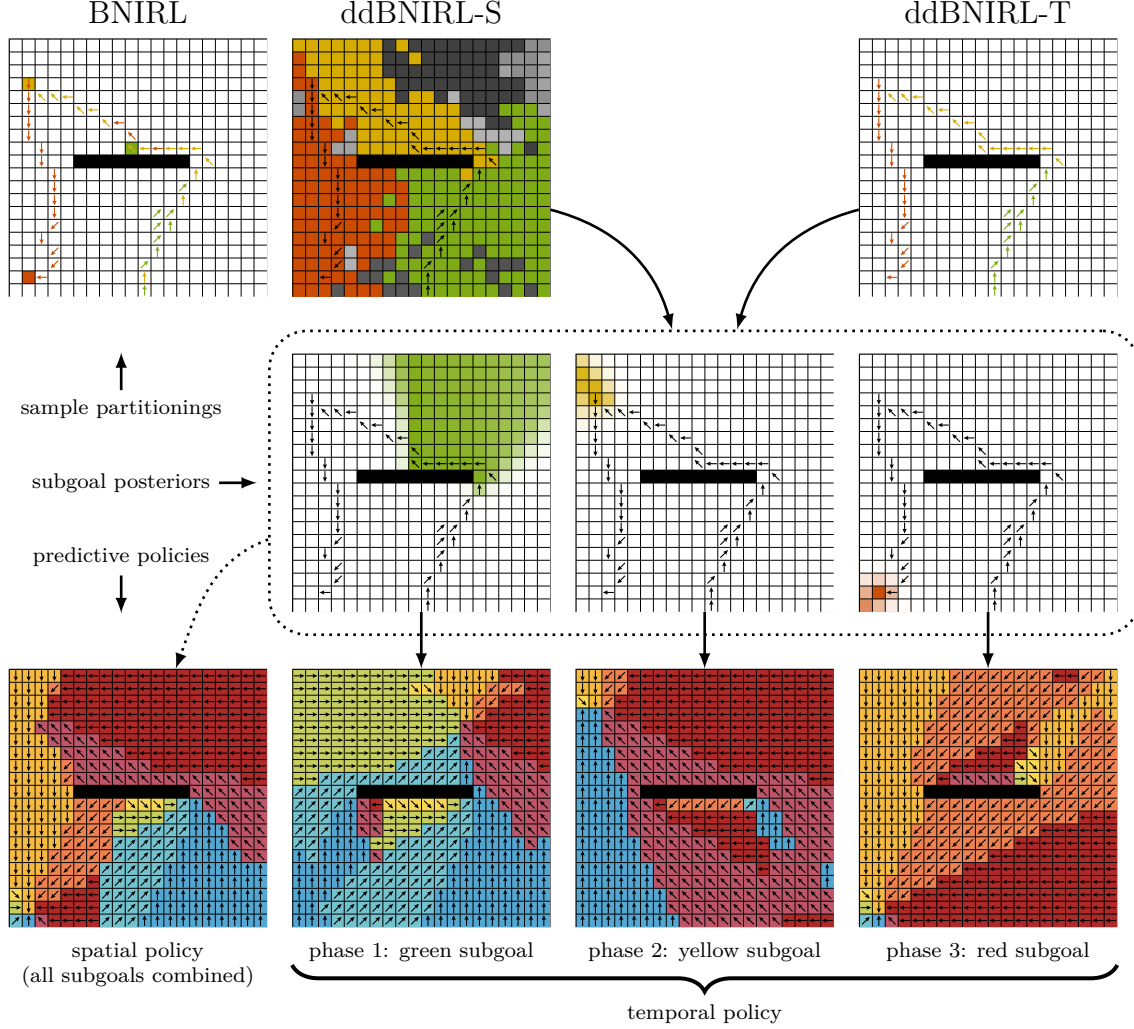


Figure 11.1: Results on the BNIRL data set [MH12]. **Top row:** Demonstration data and sample partitionings generated by the different inference algorithms. **Center row:** Subgoal posterior distributions associated with the partitions found by ddBNIRL-S and ddBNIRL-T. For a clearer overview, the corresponding BNIRL distributions are omitted (see Figure 10.1 for a comparison). **Bottom row:** Time-invariant ddBNIRL-S policy model synthesized from all three detected subgoals (left) and temporal phases identified by ddBNIRL-T (right). The background colors have no particular meaning and were added only to highlight the structures of the policies. Because of its missing generalization mechanism, BNIRL does not itself provide a reasonable predictive policy model (Limitation 1).

11.1 Proof of Concept

To illustrate the conceptual differences to BNIRL and provide additional insights into the latent intentional model learned through our framework, we begin with the motivating data set from Figure 8.1a, which had been originally presented by Michini and How [MH12]. The considered system environment, defined by $|\mathcal{S}| = 20 \times 20 = 400$ grid positions, is again shown in the top left corner of Figure 11.1. Nine of those positions correspond to inaccessible wall states, marked by the horizontal black bar. At the valid states, the expert can choose from an action set comprising a total of eight actions, each initiating a noisy state transition toward one of the (inter-)cardinal directions.

The observed state-action pairs are depicted in the form of arrows, whose colors indicate the MAP partitioning learned through BNIRL. The remaining subfigures show the results of the ddBNIRL framework, which were obtained from a posterior sample returned by the respective algorithm (ddBNIRL-S/ddBNIRL-T) at a low temperature in a simulated annealing schedule [KGV83].

Comparing the results, we observe the following main differences to the original approach:

- (i) Unlike BNIRL, the proposed framework allows to choose between a spatial and a temporal encoding of the observed task, providing the possibility to account explicitly for the type of demonstrated behavior (time-varying/time-invariant). As explained in Section 10.3, the context-unaware (yet in principle temporal) vanilla BNIRL inference scheme is still included as a special case.
- (ii) Exploiting the spatial/temporal context of the data, the ddBNIRL solution is inherently robust to demonstration noise, giving rise to notably smoother partitioning structures (top row). This effect is particularly pronounced in the case of real data, as we shall see later in Section 11.3.2.
- (iii) For each state partition/trajectory segment, we obtain an implicit representation of the associated subgoal in the form of a posterior distribution, without the need of assigning point estimates (center row). It is striking that the posterior distribution corresponding to the green state partition has a comparably large spread on the upper side of the wall. This can be explained intuitively by the fact that any subgoal located in this high posterior region could have potentially caused the green state sequence, which circumvents the wall from the right. At the same time, the green area of high posterior values exhibits a sharp boundary

on the left side since a subgoal located in the upper left region of the state space would have more likely resulted in a trajectory approaching from the left.

- (iv) In contrast to BNIRL, which has no built-in generalization mechanism (Limitation 1), our method returns a predictive policy model comprising the full posterior action information at all states. Note that we only show the resulting MAP policy estimates here (bottom row), computed according to Equation (10.8). Additional results concerning the posterior uncertainty are provided in Section 11.3.

The example illustrates how the synthesis of the predictive policy differs between ddBNIRL-S (bottom left) and ddBNIRL-T (bottom row, rightmost three subfigures). While ddBNIRL-T uses a set of (conditionally) independent policy models to describe the different identified behavioral phases, ddBNIRL-S maps the entire subgoal schedule onto a single time-invariant policy representation. Looking closer at the learned models, we recognize that the ddBNIRL-S solution in fact realizes a spatial combination of the three temporal ddBNIRL-T components, where each component is activated in the corresponding cluster region of the state space. This gives us two alternative interpretations of the same behavior.

11.2 Random MDP Scenario

Our next experiment is designed to provide insights into the generalization abilities of the framework. For this purpose, we consider a class of randomly generated MDPs similar to the Garnet problems [Bha+09]. The transition dynamics $\{\mathcal{T}(\cdot | s, a)\}$ are sampled independently from a symmetric Dirichlet distribution with a concentration parameter of 0.01, where we choose $|\mathcal{S}| = 100$ and $|\mathcal{A}| = 10$. For each repetition of the experiment, N_R states are selected uniformly at random and assigned rewards that are, in turn, sampled uniformly from the interval $[0, 1]$. All other states contain zero reward. Next, we compute an optimal deterministic MDP policy π^* with respect to a discount factor of $\gamma = 0.9$ and generate a number of expert trajectories of length 10. Herein, we let the expert select the optimal action with probability 0.9 and a random, suboptimal action with probability 0.1. The obtained state sequences are passed to the algorithms and we compute the normalized value loss of the reconstructed policies according to

$$L(\pi^*, \hat{\pi}) \triangleq \frac{\|\mathbf{V}^* - \mathbf{V}^{\hat{\pi}}\|_2}{\|\mathbf{V}^*\|_2}, \quad (11.1)$$

where \mathbf{V}^* and $\mathbf{V}^{\hat{\pi}}$ represent, respectively, the vectorized value functions of the optimal policy π^* and the reconstruction $\hat{\pi}$.

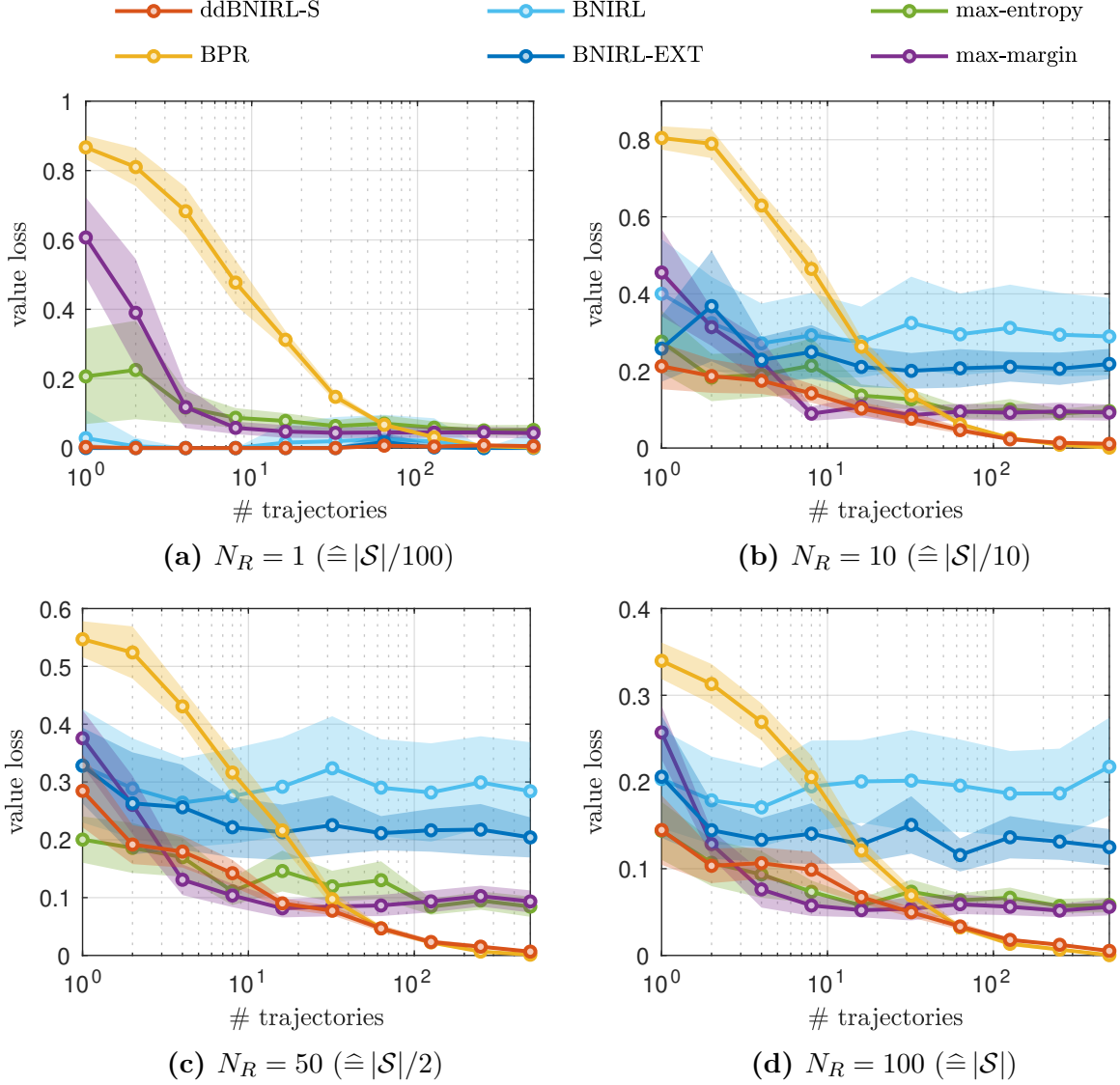


Figure 11.2: Comparison of all inference methods in the random MDP scenario for different reward densities. Shown are the empirical mean values and standard deviations of the resulting value losses, obtained from 100 Monte Carlo runs. The graphs show a clear difference between BNIRL, BNIRL-EXT and ddBNIRL-S, which illustrates the importance of considering the spatial context for subgoal extraction.

Since the considered system belongs to the class of time-invariant MDPs, ddBNIRL-S lends itself as the natural choice to model the expert behavior. As baseline methods, we adopt the subintentional Bayesian policy recognition (BPR) framework from Part I (using the ddCRP prior model described in Section 5.2), as well as max-margin IRL [AN04], max-entropy IRL [Zie+08], and vanilla BNIRL. Due to the missing generalization abilities of BNIRL (Limitation 1) and because the waypoint method (Section 9.2) does not straightforwardly apply to the considered scenario of multiple unaligned trajectories, we further compare our algorithm to an extension of BNIRL, which we refer to as BNIRL-EXT. Mimicking the ddBNIRL-S principle, the method accounts for the spatial context of the demonstrations by assigning each state to the BNIRL subgoal that is targeted by the closest (see metric in Section 10.2) state-action pair — however, these assignments are made *after* the actual subgoal inference. When compared to ddBNIRL-S, this provides a reference of how much can be gained by considering the spatial relationship of the data during the inference.

For the experiment, both ddBNIRL-S and BNIRL/BNIRL-EXT are augmented with their corresponding action sampling stages (Section 10.4.4) since the action sequences of the expert are discarded from the data set, in order to enable a fair comparison to the remaining algorithms. Figure 11.2 shows the value loss over the size of the demonstration set for different reward settings. For small N_R , both ddBNIRL-S and BNIRL/BNIRL-EXT significantly outperform the reference methods. This is because the sparse reward structure allows for an efficient subgoal-based encoding of the expert behavior, which enables the algorithms to reconstruct the policy even from minimal amounts of demonstration data.

However, the BNIRL/BNIRL-EXT solutions drastically deteriorate for denser reward structures. In particular, we observe a clear difference in performance between the cases where

- (i) we do not account for the spatial information in the partitioning model (BNIRL),
- (ii) include it in a post-processing step (BNIRL-EXT), and
- (iii) exploit it during the inference itself (ddBNIRL-S),

which demonstrates the importance of processing the context information. Most tellingly, ddBNIRL-S outperforms the baseline methods even in the dense reward regimes, although the subgoal-based encoding loses its efficiency here. In fact, the results reveal that the proposed approach combines the merits of both model types, i.e., the sample efficiency of the intentional models (max-margin IRL/max-entropy IRL)

required for small data set sizes, as well as the asymptotic accuracy and fully probabilistic nature of the subintentional Bayesian framework (BPR).*

11.3 Robot Experiment

In the next experiment, we test the ddBNIRL framework on various real data sets, which we recorded on a KUKA lightweight robotic arm (Figure 1.1b) via kinesthetic teaching. Illustrations of all demonstrated tasks are provided in Appendix B.

The system has seven degrees of freedom, corresponding to the seven joints of the arm. Each joint is equipped with a torque sensor and an angle encoder, providing recordings of joint angles, velocities and accelerations. For our experiments, we only consider the xy-Cartesian coordinates spanning the transverse plane, which we computed from the raw measurements using a forward kinematic model. The data was recorded at a sampling rate of 50 Hz and further downsampled by a factor of 10, yielding an effective sample rate of 5 Hz, which provided a sufficient temporal resolution for the considered scenario.

The goal of the experiment is to learn a set of high-level intentional models for the recorded behavior types by partitioning the data sets into meaningful parts that can be used to predict the desired motion direction of the expert. For simplicity and to demonstrate the algorithm’s robustness to modeling errors, we adopt the simplistic transition model from Section 11.1 with the same action set containing the eight (inter-)cardinal motion directions. The high measurement accuracy of the end-effector position allows us to extract these high-level actions directly from the raw data, i.e., by selecting the directions with the smallest angular deviations from the ground truth (see example in Figure 11.3a). The underlying state space is obtained by discretizing the part of the coordinate range that is covered by the measurements into blocks of predefined size (see next sections for details). Apart from this discretization step and the aforementioned data downsampling, no preprocessing is applied.

* The comparably large loss of BPR for small data set sizes can be explained by the fact that the framework is based on a more general policy model in which the expert behavior is assumed to be *inherently* stochastic (see Part I), in contrast to the here considered setting where stochasticity arises merely a consequence of suboptimal decision-making. Unlike the remaining algorithms, which already start from the assumption that the expert mimics a deterministic policy, BPR needs to infer this piece of information from the demonstration set, hence requiring a larger amount of input data.

11.3.1 Spatial Partitioning

First, we consider a case where the expert behavior can be described using a time-invariant policy model, which we aspire to capture via ddBNIRL-S. For our example, we consider the “Cycle” task shown in Appendix B. The same setting is analyzed using the time-variant ddBNIRL-T model in Section 11.3.2, which allows a direct comparison of the two approaches. The task consists in approaching a number of target positions, indicated by a set of markers (see Figure 1.1b), before eventually returning to the initial state. The setting can be regarded as a real-world version of the “Loop” problem described by Michini and How [MH12]. As explained in their paper, classical IRL algorithms that rely on a global state-based reward model (such as max-margin IRL and max-entropy IRL) completely fail on this problem, due to the periodic nature of the task.

Figure 11.3a shows the downsampled and discretized data set (black arrows) obtained from four expert trajectories (white lines). For visualization purposes, the discretization block size is chosen as $2\text{ cm} \times 2\text{ cm}$, giving rise to a total of $18 \times 24 = 432$ states. As in the top row of Figure 11.1, the coloring of the background indicates the learned partitioning structure, computed from a low-temperature posterior sample. We observe that the found state clusters clearly reveal the modular structure of the task, providing an intuitive and interpretable explanation of the data. However, although the induced policy model (Figure 11.3b) smoothly captures the cyclic nature of the task, we cannot expect to obtain trustworthy predictions in the center region of the state space, due to the lack of additional demonstration data that would be required to unveil the expert’s true intention in that region.

Clearly, a point estimate such as the shown MAP policy cannot reflect this prediction uncertainty since it does not carry any confidence information. Yet, following a Bayesian approach, we can naturally quantify the prediction uncertainty at any query state s^* based on the shape of the corresponding posterior predictive action distribution $p(a^* | s^*, \mathcal{D})$. A straightforward option is, for example, to consider the prediction entropy, defined as

$$H(s^*) \triangleq \sum_{a^* \in \mathcal{A}} p(a^* | s^*, \mathcal{D}) \log p(a^* | s^*, \mathcal{D}).$$

In order to obtain an unbiased approximation of the true *non-tempered* predictive distribution $p(a^* | s^*, \mathcal{D})$, we run a second Gibbs chain with unaltered temperature in parallel to the tempered chain. The resulting entropy estimates are summarized in an uncertainty map (Figure 11.3c), which we overlaid on the original prediction result to produce the final figure shown at the bottom right. Note that the obtained posterior

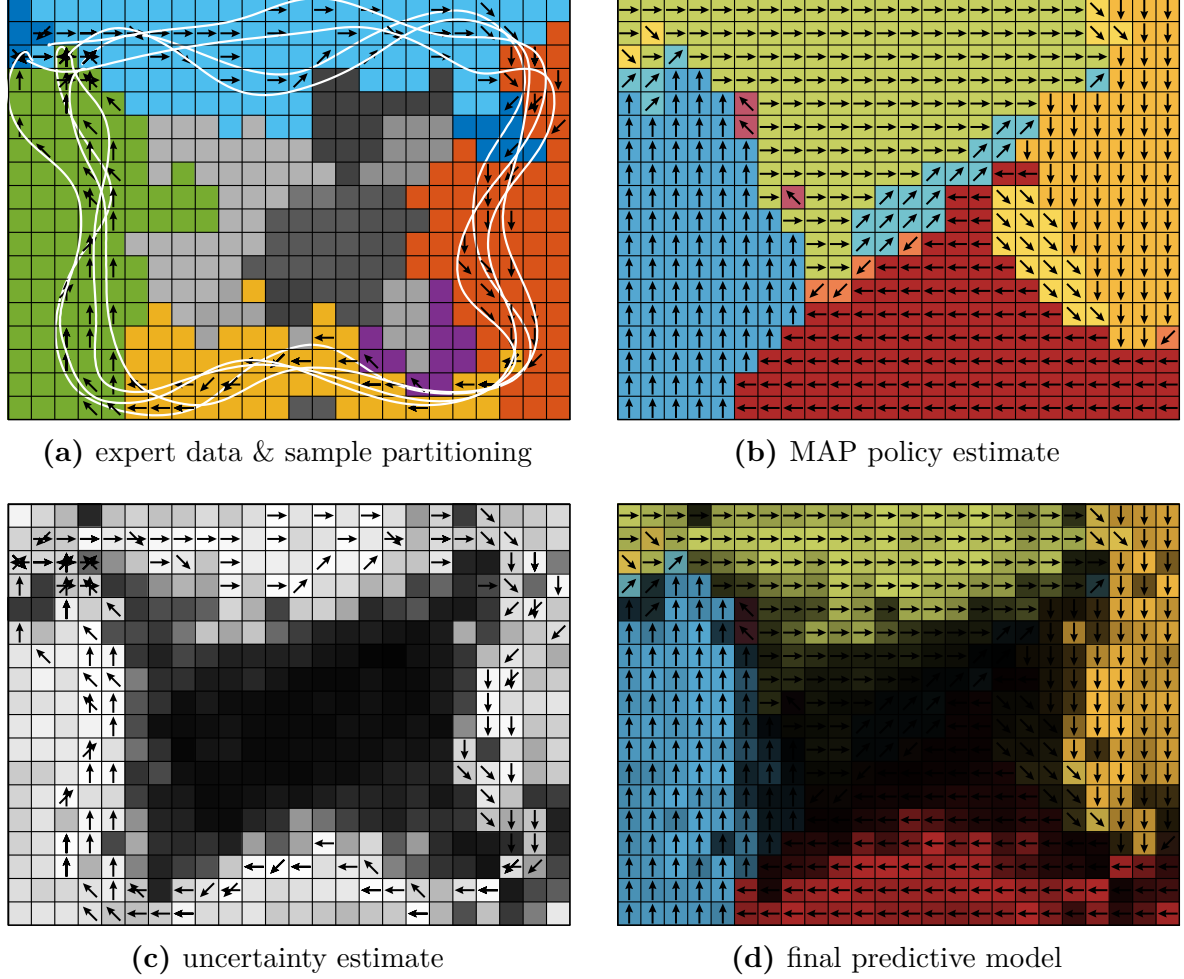


Figure 11.3: Results of ddBNIRL-S on the “Cycle” task (Appendix B). (a) Raw measurements (white lines) and discretized demonstration data (black arrows). The coloring of the background indicates a partitioning structure obtained from a low-temperature posterior sample. (b) Maximum a posteriori policy estimate. (c) Visualization of the model’s prediction uncertainty at all system states, represented by the entropies of the corresponding posterior predictive action distributions. Dark background indicates high uncertainty. (d) Illustration of the final predictive model, comprising both the action information and the prediction uncertainty.

uncertainty information of the model can be further used in an active learning setting, as demonstrated in Section 11.4.

11.3.2 Temporal Partitioning

Next, we turn our attention to the ddBNIRL-T model, which we test against the vanilla BNIRL approach. For this purpose, we consider the full collection of tasks shown in Appendix B, which comprises different time-dependent expert behaviors of varying complexity. In order to obtain a quantitative performance measure for our evaluation, we conducted a manual segmentation of all recorded trajectories, thereby creating a set of ground truth subgoal labels for all observed decision times. The result of this segmentation step is depicted in the center column of Figure B. Note that the ground truth subgoals are assumed immediately at the ends of the corresponding segments.

The left and right column of Figure B show, respectively, the partitioning structures found by BNIRL and ddBNIRL-T, based on a uniform subgoal prior distribution with support at the visited states. The underlying state discretization block size is chosen as $1\text{ cm} \times 1\text{ cm}$, as indicated by the regular grid in the background. A simple visual comparison of the learned structures reveals the clear superiority of ddBNIRL-T over vanilla BNIRL on this problem set. For our quantitative comparison, we consider the instantaneous subgoal localization errors of the two models over the entire course of a demonstration (Figure 11.4). Herein, the instantaneous localization error for a given state-action pair is measured in terms of the Euclidean distance between the grid location of the ground truth subgoal associated with the pair and the corresponding subgoal location predicted by the model. Note that the predictions of both models are based on the entire trajectory data of an experiment, considering the full posterior information after completing the demonstration. For ddBNIRL-T, which does not directly return a subgoal location estimate but instead provides access to the full subgoal posterior distribution, the error is computed with respect to the MAP subgoal locations $\{\hat{g}_k\}$, i.e.

$$\hat{g}_k \triangleq \arg \max_{g_k \in \text{supp}(p_g)} p(g_k | \mathcal{D}, \tilde{\mathbf{c}}),$$

using the ddBNIRL-T version of Equation (10.6) — see note at the beginning of Section 10.4. The black dots in the figure indicate the time instants where the ground truth annotations change. At those time instants, we observe significantly increased localization errors for both models, which can be explained by the fact that the ground truth annotation is somewhat subjective around the switching points (see labeling in Figure B). Also, we notice a comparably high error at the beginning and the end of some

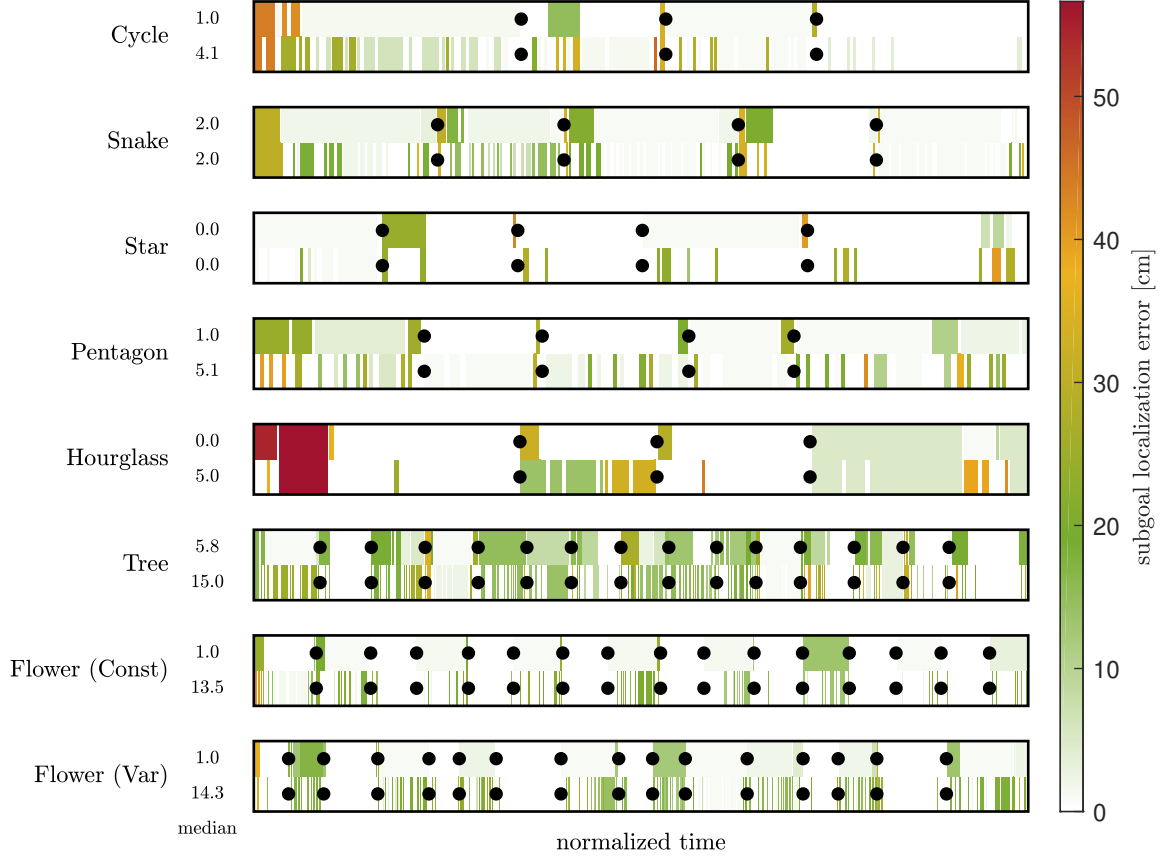


Figure 11.4: Instantaneous subgoal localization errors of ddBNIRL-T (upper rows) and BNIRL (lower rows) for the eight recorded data sets. The black dots indicate the subgoal switching times in the corresponding ground truth subgoal annotation, depicted in the center column of Figure B. On average, the localization error of ddBNIRL-T is significantly lower compared to the BNIRL approach, as indicated by the median values shown on the left. For a qualitative comparison of the underlying partitioning structures, see Appendix B.

trajectories, which stems from the imperfect synchronization between the recording interval and the execution of the task (recall that we skipped the corresponding data preprocessing step). Hence, to capture the accuracy in a single figure of performance, we consider the median localization error of each time series, as it masks out these outliers and provides a more realistic error quantification than the sample mean. The obtained values are shown next to the error plots in Figure 11.4, indicating that the ddBNIRL-T localization error is in the range of the discretization interval in most cases. Compared to BNIRL, the proposed method yields an error reduction of more than 70% on average.

11.4 Active Learning

In Section 11.3, we saw that the posterior predictive action distribution $p(a^* | s^*, \mathcal{D})$ provides a natural way to quantify the prediction uncertainty of our model at any given query state s^* . This offers the opportunity to apply the framework in an active learning setting, since the induced uncertainty map (see example in Figure 11.3c) indicates in which parts of the state space the trained model can process further instructions from the expert most effectively.

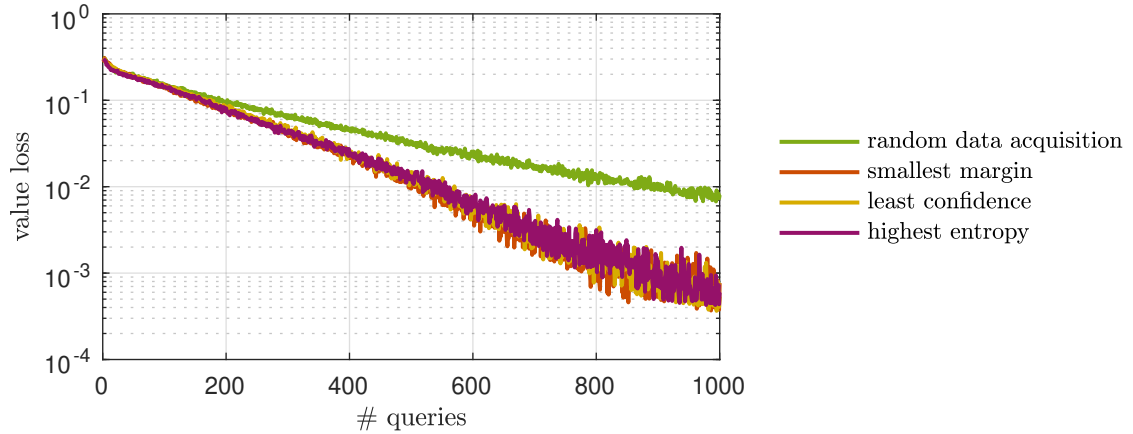


Figure 11.5: Comparison between random data acquisition and active learning in the random MDP scenario. Shown are the empirical mean value losses of the obtained policy models over the number of data queries, obtained from 200 Monte Carlo runs.

To demonstrate the basic procedure, we reconsider the random MDP problem from Section 11.2 in an active learning context, where we compare different active strategies with the previously used random data acquisition scheme. As an initialization for the learning procedure, we request a single state-action pair (s_1, a_1) from the demonstrator,

which we store in the initial data set $\mathcal{D}_1 \triangleq \{(s_1, a_1)\}$. Herein, the state s_1 is drawn uniformly at random from \mathcal{S} and the action $a_1 \sim \pi_E(a | s_1)$ is generated according to the noisy expert policy $\pi_E : \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ described in Section 11.1. Continuing from this point, each of the considered active learning algorithms requests a series of subsequent demonstrations $((s_2, a_2), (s_3, a_3), \dots)$, inducing a sequence of data sets $(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \dots)$, where the next query state s_{d+1} is chosen according to the specific data acquisition criterion f_{acq} of the algorithm evaluated on the current predictive model, i.e.

$$\begin{aligned}\mathcal{D}_d &= \mathcal{D}_{d-1} \cup \{(s_{d+1}, a_{d+1})\}, \\ s_{d+1} &= \arg \max_{s^* \in \mathcal{S}} f_{\text{acq}}[p(a^* | s^*, \mathcal{D}_d)], \\ a_{d+1} &\sim \pi_E(a | s_{d+1}).\end{aligned}$$

The purpose of the acquisition criterion is to assess the uncertainty of the model at all possible query states, so that the next demonstration can be requested in the high uncertainty region of the state space (see *uncertainty sampling* [Set12]). For our experiment, we consider the following three common choices,

- highest entropy: $f_{\text{acq}}(p) \triangleq - \sum_{a \in \mathcal{A}} p(a) \log p(a)$,
- least confidence: $f_{\text{acq}}(p) \triangleq 1 - \max_{a \in \mathcal{A}} p(a)$,
- smallest margin: $f_{\text{acq}}(p) \triangleq p(\hat{a}_2) - p(\hat{a}_1)$,

where \hat{a}_1 and \hat{a}_2 denote, respectively, the most likely and second most likely action according to the considered distribution p , i.e.,

$$\hat{a}_1 \triangleq \arg \max_{a \in \mathcal{A}} p(a) \quad \text{and} \quad \hat{a}_2 \triangleq \arg \max_{a \in \mathcal{A} \setminus \hat{a}_1} p(a).$$

At each iteration, we compute the value losses (Equation 11.1) of the induced policy models and compare them with the corresponding loss obtained from random data acquisition. The resulting curves are delineated in Figure 11.5. As expected, the learning speed of the model is significantly improved under all active acquisition schemes, which reduces the number of expert demonstrations required to successfully learn the observed task.

12

Summary

Building upon the principle of Bayesian nonparametric inverse reinforcement learning, we proposed a new framework for data-efficient IRL that leverages the context information of the demonstration set to learn a predictive model of the expert behavior from small amounts of training data. Central to our framework are two model architectures, one designed for learning spatial subgoal plans, the other to capture time-varying intentions. In contrast to the original BNIRL model, both architectures explicitly consider the covariate information contained in the demonstration set, giving rise to predictive models that are inherently robust to demonstration noise. While the original BNIRL model can be recovered as a special case of our framework, the conducted experiments show a drastic improvement over the vanilla BNIRL approach in terms of the achieved subgoal localization accuracy, which stems from both an improved likelihood model and a context-aware clustering of the data. Most notably, our framework outperforms all tested reference methods in the analyzed benchmark scenarios while it additionally captures the full posterior information about the learned subgoal representation. The resulting prediction uncertainty about the expert behavior, reflected by the posterior predictive action distribution, provides a natural basis to apply our method in an active learning setting, where the learning system can request additional demonstration data from the expert.

PART III

Multi-Agent Systems: Swarm Modeling

13 Motivation	117
14 The swarMDP Model	119
15 Inverse Reinforcement Learning in Swarm Systems	121
16 Experimental Results	130
17 Summary	136

With this part of the thesis, we move over to the multi-agent domain and focus our attention to *swarm systems*—large-scale populations of agents based on a single agent prototype. Our goal is to transfer the IRL principle to the swarm scenario, with the motivation to create a computational framework that can be used to gain insights into the mechanics of self-organizing distributed systems. To this end, we introduce a new model class of agent networks that compactly encodes the homogeneity properties of our target systems and finally provides the mathematical basis for a generalized inference scheme.

13

Motivation

Emergence and the ability of self-organization are fascinating characteristics of natural systems with interacting agents. Without a central controller, these systems are inherently robust to failure while, at the same time, they show remarkable large-scale dynamics that allow for fast adaptation to changing environments [Buh+06; Cou09]. Interestingly, for large system sizes, it is often not the complexity of the individual agent but the (local) coupling of the agents that predominantly gears the system dynamics. It has been shown, in fact, that even relatively simple local dynamics can lead to various

kinds of higher-order complexity at a global scale when coupled through a network with many agents [OMT08; VZ12].

Unfortunately, the complex relationship between the global behavior of a system and its local implementation at the agent level is not well understood. In particular, it remains unclear when — and how — a global system objective can be encoded in terms of local rules, and what are the requirements on the complexity of the individual agent in order for the collective to fulfill a certain task. Yet, a thorough understanding of this relationship is key to advancement in many technologies like swarm robotics [Bra+13], large-scale sensor networks [LOT03], nanomedicine [Fre05], programmable matter [GCM05] and self-assembly systems [WG02].

By providing a data-driven approach to behavioral modeling, IRL offers a promising tool that can help to establish this missing link. Unfortunately, most of the IRL research has been dedicated to the single-agent case and its capabilities for multi-agent modeling are mostly unexplored. Recent work on multi-agent IRL largely focuses on two-agent scenarios [Had+16; LBC14], and there exist only few approaches that transfer the IRL concept to systems with several agents (see [KS15] for a game-theoretic view on this problem). A notable exception is the work by Natarajan et al. [Nat+10], which extends the IRL principle to non-cooperative multi-agent domains in order to learn a joint reward model that is able to explain the observed system behavior at a global scale. Yet, the authors assume that all agents in the network are controlled by a central mediator, which makes the framework unsuitable for application to self-organizing systems. A decentralized solution was later presented by Reddy et al. [Red+12] but the proposed algorithm is based on the simplifying assumption that all agents are informed about the global system state. Finally, Dufton and Larson [DL09] presented a multi-agent framework based on mechanism design, which can be used to refine a given reward model in order to promote a certain system behavior. However, the framework is not able to learn the reward model entirely from demonstration data.

In contrast to the existing literature, the methodology presented in this part of the dissertation focuses specifically on *large-scale modeling*. Motivated by idea of uncovering the latent system objective in a swarm network, we present a scalable IRL framework to learn a single *local* reward function that explains the *global* behavior of the system. As part of this framework, we propose a reinforcement learning scheme that allows to reconstruct the swarm behavior from local interactions at the agent level. We begin with a formal definition of swarm systems and a discussion of the relevant characteristics.

14

The swarMDP Model

By analogy with the characteristics of natural agent populations (Figure 1.1a), we define a *swarm system* as a collection of agents with the following two properties:

- **Homogeneity:** All agents in a swarm share a common architecture, i.e., they have the same dynamics, degrees of freedom and observation capabilities. Accordingly, all agents are assumed to be interchangeable.
- **Locality:** The agents can only observe (and interact with) parts of the system in their vicinity, as determined by their observation capabilities. This implies that their decisions depend on their local neighborhood only and not on the whole swarm state.

In principle, any system with such properties can be described as a decentralized partially observable Markov decision process (dec-POMDP) [Oli12]. However, the homogeneity property, which turns out to be the key ingredient for scalable inference, is not explicitly captured by this model class. Because the number of agents contained in a swarm is typically large, it is convenient to switch to a more compact system representation that exploits the system symmetries.

For this reason, we introduce a new sub-class of dec-POMDP models, in the following referred to as swarMDPs, which explicitly implement a homogeneous agent architecture (Figure 14.1). An agent in this model, which we call a *swarming agent*, is defined as a tuple $\mathbb{A} \triangleq (\mathcal{S}, \mathcal{O}, \mathcal{A}, R, \pi)$, where

- \mathcal{S} , \mathcal{O} and \mathcal{A} are sets of local states, observations and actions, respectively.
- $R : \mathcal{O} \rightarrow \mathbb{R}$ is an agent-level reward function.
- $\pi : \mathcal{O} \rightarrow \mathcal{A}$ is the local policy of the agent, which later serves as the decentralized control law of the swarm.

For the sake of simplicity, we consider only reactive policies in this work (see Section 2.1.2), where π is a function of the agent's current observation. Note, however, that an extension to more general policy models (e.g., belief state policies [KLC98] or such that operate on observation histories [Oli12]) is straightforward.

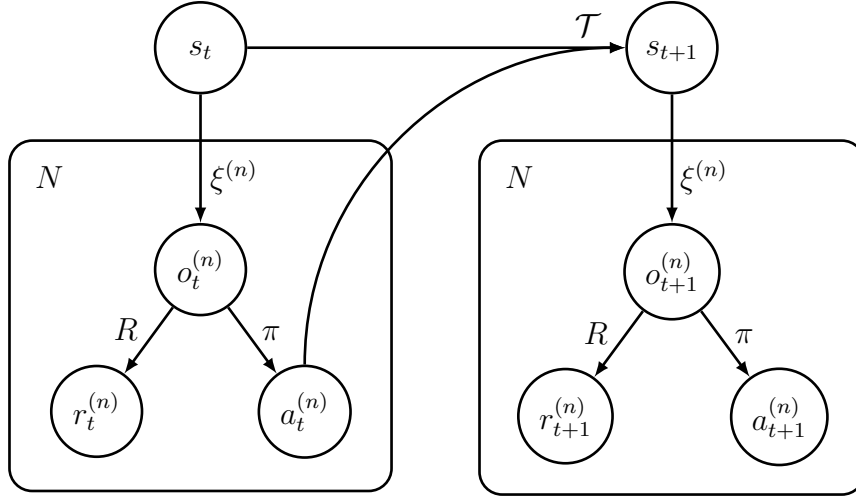


Figure 14.1: The swarMDP model visualized as a Bayesian network. In contrast to a dec-POMDP, the model explicitly encodes the homogeneity of a swarm system.

With the definition of the swarming agent at hand, we now define a swarMDP as a tuple $(N, \mathbb{A}, \mathcal{T}, \xi)$, where

- N is the number of agents in the system.
- \mathbb{A} is a swarming agent prototype as defined above.
- $\mathcal{T} : \mathcal{S}^N \times \mathcal{A}^N \times \mathcal{S}^N \rightarrow \mathbb{R}$ is the global transition model of the system. While used only implicitly later on, it defines the conditional probability of the event that the system reaches the global state $\tilde{s} = (\tilde{s}^{(1)}, \dots, \tilde{s}^{(N)})$ when the agents perform the joint action $a = (a^{(1)}, \dots, a^{(N)})$ at state $s = (s^{(1)}, \dots, s^{(N)})$, which we denote by $\mathcal{T}(\tilde{s} | s, a)$. Herein, $s^{(n)}, \tilde{s}^{(n)} \in \mathcal{S}$ and $a^{(n)} \in \mathcal{A}$ represent the local states and the local action of agent n , respectively.
- $\xi : \mathcal{S}^N \rightarrow \mathcal{O}^N$ is the global observation model of the system.

The observation model ξ tells us which parts of a given system state $s \in \mathcal{S}^N$ can be observed by whom. More precisely, $\xi(s) = (\xi^{(1)}(s), \dots, \xi^{(N)}(s)) \in \mathcal{O}^N$ denotes the ordered set of local observations passed on to the agents at state s , i.e., agent n receives observation $\xi^{(n)} \in \mathcal{O}$. For example, in a school of fish, $\xi^{(n)}$ could represent the local alignment of agent n to its immediate neighbors (see Section 16.1). Note in particular that the agent has no access to its local states $s^{(n)} \in \mathcal{S}$ but only to its local observations $o^{(n)} \triangleq \xi^{(n)}(s)$. A generalization to stochastic observations is possible but not considered here.

Remark 1 (Homogeneity). It should be mentioned that the observation model ξ could be alternatively defined locally at the agent level, since the observations $\{o^{(n)}\}$ are

agent-related quantities. However, this would still require a global notion of connectivity between the agents, e.g., provided in the form of a dynamic graph that defines the time-varying neighborhood of the agents. Using a global observation model, we can encode all properties into a single object, yielding a more compact system description. Yet, we need to constrain our observation model class to those models that respect the homogeneity (and thus the interchangeability) of the agents. To be precise, a valid swarm observation model needs to ensure that agent n receives agent m 's local observation (and vice versa) if we interchange their local states. Mathematically, this means that any permutation of the system state $s \in \mathcal{S}^N$ must result in the same permutation of $\xi(s)$ —otherwise, the underlying system is not homogeneous. Similarly, the transition model is required to be permutation invariant, i.e., it has to hold that $\mathcal{T}(\sigma(\tilde{s}) \mid \sigma(s), \sigma(a)) = \mathcal{T}(\tilde{s} \mid s, a)$ for any permutation operator σ .

————— 15 ————— Inverse Reinforcement Learning in Swarm Systems

In contrast to most existing work on IRL, we do *not* intent to devise a new specialized algorithm that solves the IRL problem in the swarm case. Instead, we show that the homogeneity of the swarMDP model allows us to reduce the multi-agent IRL problem to a single-agent one, for which a whole suite of existing algorithms can be applied. This reduction is possible since, at its heart, the underlying optimization problem intrinsically remains a single-agent control problem, as all agents in the system are interchangeable.

In the subsequent sections, we show that the inherent symmetry property of the system also translates to the value functions of the agents, which allows a straightforward generalization of the IRL principle to the swarm setting. Algorithmically, we exploit the fact that most existing IRL methods, such as [AN04; NS07; NR00; RA07; SS08; Zie+08], share a common generic form (Algorithm 1), which involves three main steps [MH12]: 1) policy update 2) value estimation and 3) reward update. The important detail to note is that only the first two steps of this procedure are system-specific while the third step is, in fact, independent of the target system (see references listed above for algorithmic details). Consequently, our problem reduces to finding swarm-based solutions for the first two steps such that the overall procedure returns a meaningful reward model in the IRL context. The following sections discuss these steps in detail.

Algorithm 1: GENERIC IRL

Input: expert data \mathcal{D} , system model (MDP\R)

Output: sequence of reward function estimates $R^{\{1\}}, R^{\{2\}}, \dots$

0: Initialize reward function $R^{\{0\}}$

for $i = 0, 1, 2, \dots$

1: Policy update: Find an optimal policy $\pi^{\{i\}}$ for $R^{\{i\}}$

2: Value estimation: Estimate the corresponding value $V^{\{i\}}$

3: Reward update: Given $V^{\{i\}}$ and \mathcal{D} , compute next reward estimate $R^{\{i+1\}}$

15.1 Policy Update

We start with the policy update step, which poses the problem of learning an optimal system policy for a given reward function. For this purpose, it is first necessary to define a suitable learning objective for the swarm setting. In the following, we show that the homogeneity property of our model naturally induces such an objective, and we furthermore present a simple learning strategy for optimization.

15.1.1 Private Value and Bellman Optimality

Analogous to the single-agent case (Section 2.1.3), we define the *private value* of agent n at a swarm state $s \in \mathcal{S}^N$ under policy π as the expected sum of discounted rewards accumulated by the agent over time, given that all agents execute π , i.e.

$$V^{(n)}(s | \pi) \triangleq \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k R(\xi^{(n)}(s_{t+k})) \mid \pi, s_t = s \right], \quad (15.1)$$

where the expectation is with respect to the random system trajectory starting from s . In contrast to the single-agent setting in Equation (2.1), we herein explicitly consider the system at a particular reference time t , for reasons that will become clear in Section 15.1.2. Note, however, that the above value definition is independent of the particular choice of t , due to the assumed time-homogeneity of the transition model \mathcal{T} ; hence, the symbol t vanishes from the left hand side of the equation.

Denoting further by $Q^{(n)}(s, a | \pi)$ the state-action value of agent n at state s for the case that all agents execute policy π —except for agent n , who performs action $a \in \mathcal{A}$

once and follows π thereafter—we obtain the following Bellman equations,

$$\begin{aligned} V^{(n)}(s | \pi) &= R(\xi^{(n)}(s)) + \gamma \sum_{\tilde{s} \in \mathcal{S}^N} p(\tilde{s} | s, \pi) V^{(n)}(\tilde{s} | \pi), \\ Q^{(n)}(s, a | \pi) &= R(\xi^{(n)}(s)) + \gamma \sum_{\tilde{s} \in \mathcal{S}^N} p^{(n)}(\tilde{s} | s, a, \pi) V^{(n)}(\tilde{s} | \pi). \end{aligned}$$

Herein, $p(\tilde{s} | s, \pi)$ denotes the probability of reaching swarm state \tilde{s} from s when *every* agent performs policy π and, analogously, $p^{(n)}(\tilde{s} | s, a, \pi)$ denotes the probability of reaching swarm state \tilde{s} from state s if agent n chooses action a and *all other* agents execute policy π . Note that both probabilities are defined implicitly via the transition model \mathcal{T} .

15.1.2 Local Value

Unlike in the fully observable setting described in Section 2.1, the value function in Equation (15.1) is not locally plannable by the agents since they have no access to the global swarm state s . In order to define a suitable learning objective for the system, we require an alternative notion of optimality that is solely based on local information and, hence, computable by the agents.

Analogous to the belief value in single-agent systems [Mel08; Meu+99], we therefore introduce an additional *local value function*, which reflects the expected return of agent n in consideration of its current local observation of the global system state, i.e.

$$V_t^{(n)}(o | \pi) \triangleq \mathbb{E}_{p_t(s | o^{(n)}=o, \pi)} [V^{(n)}(s | \pi)], \quad (15.2)$$

The following proposition highlights two key properties of this quantity: (i) it is not only locally plannable but also reduces the multi-agent problem to a single-agent one, in the sense that all local values coincide. (ii) Unlike the private value in Equation (15.1), the local value is time-dependent because the conditional probabilities $p_t(s | o^{(n)}=o, \pi)$, in general, depend on time. However, it converges to a stationary value asymptotically under suitable conditions.

Proposition 2 (Stationary value). *Consider a swarMDP as defined in Chapter 14 and the stochastic process $\{s_t\}_{t=0}^\infty$ of the swarm state induced by the system policy π . If the initial state distribution of the system is invariant under permutation* of agents, all*

* Since we assume that the agents are interchangeable, it follows naturally to consider only permutation-invariant initial distributions.

local value functions are identical, i.e.

$$V_t^{(m)}(o | \pi) = V_t^{(n)}(o | \pi) \quad \forall m, n.$$

In this case, we may drop the agent index and denote the common local value function as $V_t(o | \pi)$. If, furthermore, it holds that $s_t \xrightarrow{a.s.} s$ for some s with law p and the common local value function is continuous almost everywhere (i.e., its set of discontinuity points is p -null) and bounded above, then the local value function $V_t(o | \pi)$ will converge to a limit, i.e.

$$V_t(o | \pi) \rightarrow V(o | \pi), \quad (15.3)$$

where $V(o | \pi) = \mathbb{E}_{p(s | o^{(n)}=o, \pi)} [V^{(n)}(s | \pi)]$.

Proof. Fix any two agents, say agent 1 and agent 2. For these agents, define a permutation operation $\sigma : \mathcal{S}^N \rightarrow \mathcal{S}^N$ as

$$\sigma(s) \triangleq (s^{(2)}, s^{(1)}, s^{(3)}, \dots, s^{(N)}),$$

where $s = (s^{(1)}, s^{(2)}, s^{(3)}, \dots, s^{(N)})$. Due to the homogeneity of the system, i.e., since $R(\xi^{(1)}(s)) = R(\xi^{(2)}(\sigma(s)))$ and $p(\tilde{s} | s, \pi) = p(\sigma(\tilde{s}) | \sigma(s), \pi)$, it follows immediately that $V^{(1)}(s | \pi) = V^{(2)}(\sigma(s) | \pi) \forall s$. This essentially means that the private value assigned to agent 1 at swarm state s is identical to the value that would be assigned to agent 2 if we interchanged their local states, i.e., at state $\sigma(s)$. Note that this is effectively the same as renaming the agents. The homogeneity of the system ensures that the symmetry of the initial state distribution $p_0(s)$ is maintained at all subsequent points in time, i.e., $p_t(s | \pi) = p_t(\sigma(s) | \pi) \forall s, t$. In particular, it holds that $p_t(s | o^{(1)} = o, \pi) = p_t(\sigma(s) | o^{(2)} = o, \pi) \forall s, t$ and, accordingly,

$$\begin{aligned} V_t^{(1)}(o | \pi) - V_t^{(2)}(o | \pi) &= \mathbb{E}_{p_t(s | o^{(1)}=o, \pi)} [V^{(1)}(s | \pi)] - \mathbb{E}_{p_t(s | o^{(2)}=o, \pi)} [V^{(2)}(s | \pi)] \\ &= \sum_{s \in \mathcal{S}^N} \left(p_t(s | o^{(1)} = o, \pi) V^{(1)}(s | \pi) - p_t(\sigma(s) | o^{(2)} = o, \pi) V^{(2)}(\sigma(s) | \pi) \right) = 0, \end{aligned}$$

which shows that the local value functions are identical for all agents. Treating the value as a random variable and using the fact that it is continuous almost everywhere, it follows that $V^{(n)}(s_t | \pi) \xrightarrow{a.s.} V^{(n)}(s | \pi)$ since $s_t \xrightarrow{a.s.} s$. As we assume the function to be finite, i.e., $|V^{(n)}(s_t | \pi)| < V^*$ for some $V^* \in \mathbb{R}$, it holds by the conditional dominated convergence theorem [Bil99] that $\mathbb{E} [V^{(n)}(s_t | \pi) | o^{(n)} = o, \pi] \rightarrow \mathbb{E} [V^{(n)}(s | \pi) | o^{(n)} = o, \pi]$, i.e., $V_t(o | \pi) \rightarrow V(o | \pi)$. \square

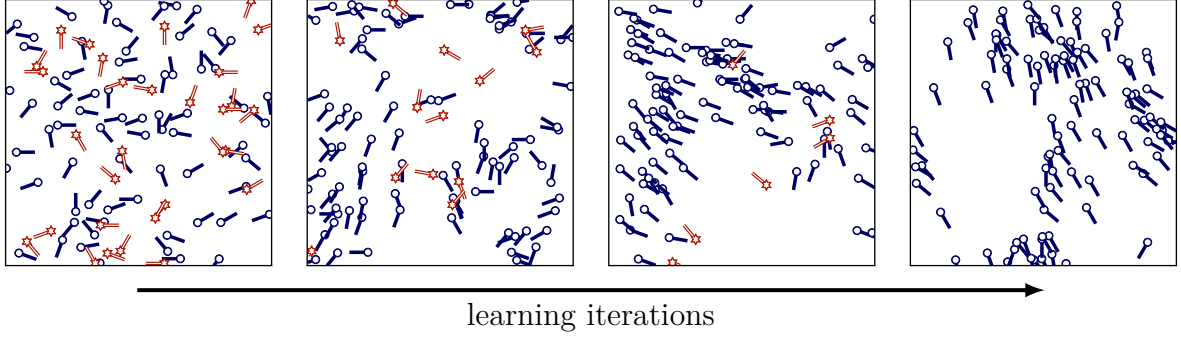


Figure 15.1: Snapshots of the proposed heterogeneous learning scheme applied to the Vicsek system (Section 16.1). The agents are divided into a greedy set (\bullet) and an exploration set (\star) to facilitate the exploration of locally desynchronized swarm states. The size of the exploration set is reduced over time to gradually transfer the system into a homogeneous stationary behavior.

15.1.3 Heterogeneous Q-learning

With the local value in Equation (15.2), we have introduced a system-wide performance measure that can be evaluated at the agent level and may, hence, be used by the agents for local planning. Yet, its computation involves the evaluation of an expectation with respect to the current swarm state of the system. This requires the agents to maintain a belief about the global state configuration at any point in time in order to coordinate their actions, which is itself a challenging problem. However, for many swarm-related tasks, like consensus problems [RBA05], we are primarily interested in the stationary behavior of the system. In these cases, the policy optimization becomes substantially simpler since it allows us to forget about the temporal aspects of the problem.

In this section, we present a comparably simple learning method, specifically tailored to the swarm setting, which solves the policy update in Algorithm 1 by optimizing the system's stationary value (Equation 15.3). Analogous to the local value function in Equation (15.2), we define a *local Q-function* for each agent, i.e.

$$Q_t^{(n)}(o, a | \pi) \triangleq \mathbb{E}_{p_t(s | o^{(n)}=o, \pi)} [Q^{(n)}(s, a | \pi)],$$

which assesses the quality of a particular action played by agent n at time t . Following the same line of argument as before, one can show that these Q-functions are also identical for all agents. Moreover, they converge to the following asymptotic value function,

$$Q(o, a | \pi) = \mathbb{E}_{p(s | o^{(n)}=o, \pi)} [Q^{(n)}(s, a | \pi)], \quad (15.4)$$

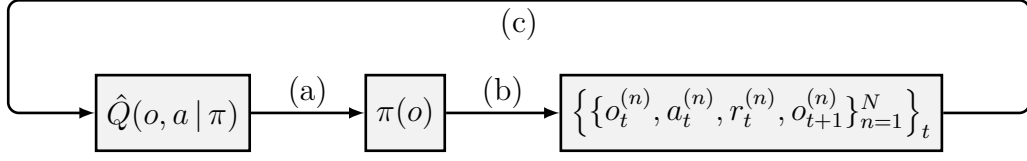


Figure 15.2: Pictorial description of the proposed heterogeneous learning scheme. **(a)** The policy at the next iteration is obtained as the best response to the current estimate of the system’s stationary Q-function. **(b)** Heterogeneous one-step transition of the system according to Algorithm 2. **(c)** The Q-function estimate is updated based on the new experience.

which can be understood as the state-action value of a generic agent that is coupled to a stationary field generated by and executing policy π .

In the following, we pose the task of optimizing this Q-function as a game-theoretic one. To be precise, we consider a hypothetical game between each agent and the environment surrounding it, where the agent plays the optimal response to the stationary field, i.e.

$$\pi_R(o | \pi) \triangleq \arg \max_{a \in A} Q(o, a | \pi), \quad (15.5)$$

and where the environment reacts with a new swarm behavior generated by that policy. By definition, any optimal system policy π^* describes a fixed point of this game, i.e.

$$\pi_R(o | \pi^*) = \pi^*(o),$$

which motivates the following iterative learning scheme: starting with an arbitrary initial policy, we run the system until it reaches its stationary behavior and estimate the corresponding asymptotic Q-function. Based on this Q-function, we update the system policy according to the best response operator defined in Equation (15.5). The updated policy, in turn, induces a new swarm behavior for which we estimate a new Q-function, and so on. As soon as we reach a fixed point, the system has arrived at an optimal solution in the form of a symmetric Nash equilibrium where all agents collectively execute a policy that, for each agent individually, provides the optimal response to the other agents’ behavior.

However, the following practical problems remain: (i) in general, it can be time-consuming to wait for the system to reach its stationary behavior at each iteration of the algorithm. (ii) At stationarity, we need a way to estimate the corresponding stationary Q-function. Note in particular that this involves both estimating the Q-values of actions that are dictated by the current policy as well as the Q-values of all actions that deviate from the current behavior, which requires exploratory moves. As

Algorithm 2: HETEROGENEOUS Q-LEARNING

Input: swarMDP without policy π
Output: estimate of stationary Q-function

- 0: Initialize shared Q-function, learning rate, and fraction of exploring agents
 for $i = 0, 1, 2, \dots$
 - 1: Divide the swarm into greedy and exploring agents
 - 2: Select actions for all agents based on current Q-function/exploration strategy
 - 3: Iterate the system and collect rewards for all agents
 - 4: Update the Q-function based on the new experience
 - 5: Decrease the learning rate and the fraction of exploring agents
-

a solution to both problems, we propose the following *heterogeneous learning scheme*, which artificially breaks the symmetry of the system by separating the agents into two disjoint groups: a greedy set and an exploration set. While the agents in the greedy set provide a reference behavior in form of the optimal response to the current Q-function estimate shared between all agents, the agents in the exploration set randomly explore the effects of different actions in the context of the current system behavior. At each iteration, the gathered experience of all agents is processed sequentially via the following Q-update (compare Equation 2.10),

$$\hat{Q}(o_t^{(n)}, a_t^{(n)}) \leftarrow (1 - \alpha_t) \hat{Q}(o_t^{(n)}, a_t^{(n)}) + \alpha_t \left(r_t^{(n)} + \gamma \max_{a \in \mathcal{A}} \hat{Q}(o_{t+1}^{(n)}, a) \right),$$

with some learning rate $\alpha_t \in (0, 1)$. Over time, more and more exploring agents are assigned to the greedy set so that the system is gradually transferred into a *homogeneous stationary regime* and thereby smoothly guided towards a fixed-point policy (see Figure 15.1). Herein, the learning rate α naturally reduces the influence of experience acquired at early (non-synchronized) stages of the system, which allows us to update the system policy without having to wait until the swarm has converged to its stationary behavior in each iteration.

The heterogeneity of the system during the learning phase ensures that also locally desynchronized swarm states are well-explored so that the agents can learn adequate responses to out-of-equilibrium situations. This phenomenon is illustrated by the agent constellation in third subfigure of Figure 15.1: it shows a situation that is highly unlikely under a homogeneous learning scheme as it requires a series of consecutive exploration steps by only a few agents while, at the same time, all their neighbors need to behave consistently optimally. The final learning procedure, which can be interpreted as a model-

free variant of policy iteration [LP03] in a non-stationary environment, is summarized in Algorithm 2. A pictorial description of the main steps is provided in Figure 15.2.

15.2 Value Estimation

In Section 15.1, we showed a simple way to implement the policy update step in Algorithm 1 based on local information gathered at the agent level. Next, we need to think about a suitable performance measure that allows a comparison between the learned behavior and the expert solution, in order to adjust the reward model in the last step of the algorithm.

15.2.1 Global Value

The comparison of the learned behavior and the expert behavior should take place on the swarm level, since we want the updated reward function to induce a new system behavior that mimics the expert behavior *globally*. For this reason, we introduce the following *global value*,

$$V_{|\pi}^{(n)} \triangleq \mathbb{E}_{p_0(s)} \left[V^{(n)}(s | \pi) \right],$$

which reflects the expected return of an agent under π , averaged over all possible initial states of the swarm. From the system symmetry, i.e., since $p_0(s) = p_0(\sigma(s))$, it follows immediately that this global value is independent of the specific agent under consideration, i.e., it holds that

$$V_{|\pi}^{(m)} = V_{|\pi}^{(n)} \quad \forall m, n.$$

Thus, the global value provides indeed a system-wide performance measure (as opposed to an agent-specific property) that may be used as a quality indicator for the reward update in the last step of Algorithm 1. We can construct an unbiased estimator for this quantity from any local agent trajectory, i.e.

$$\hat{V}_{|\pi}^{(n)} = \sum_{t=0}^{\infty} \gamma^t R(\xi^{(n)}(s_t)) = \sum_{t=0}^{\infty} \gamma^t r_t^{(n)}. \quad (15.6)$$

Since all local estimators follow the same distribution, we can obtain an estimate of reduced variance by considering the information of the whole swarm, i.e.

$$\hat{V}_{|\pi} = \frac{1}{N} \sum_{n=1}^N \hat{V}_{|\pi}^{(n)} = \frac{1}{N} \sum_{n=1}^N \sum_{t=0}^{\infty} \gamma^t r_t^{(n)}. \quad (15.7)$$

Note, however, that the local estimators in Equation (15.6) are correlated since the agents are coupled through the system process. Yet, due to the specific coupling structure of a swarm caused by the locality property (Chapter 14), correlation is introduced only *locally*, meaning that the coupling between any two agents drops when their topological distance increases. We analyze this phenomenon for the Vicsek model [Vic+95] in Section 16.1.

15.3 Reward Update

The last step of Algorithm 1 consists of updating the estimated agent reward function. Depending on the single-agent IRL framework in use, this involves an algorithm-specific optimization procedure, e.g., given in the form of a quadratic program [AN04; NR00] or a gradient-based optimization [NS07; Zie+08]. For our experiments in Chapter 16, we adopt the max-margin principle presented in [AN04]; however, the estimation procedure can be replaced with any other value-based IRL method, as explained at the beginning of this chapter.

Following the max-margin approach, the local reward function is represented as a linear combination of observational features, i.e., $R(o) = w^\top \phi(o)$, with weights $w \in \mathbb{R}^d$ and a given feature function $\phi : \mathcal{O} \rightarrow \mathbb{R}^d$. The feature weights after the i th iteration of Algorithm 1 are then obtained as

$$w^{\{i+1\}} = \arg \max_{w: \|w\|_2 \leq 1} \min_{j \in \{1, \dots, i\}} w^\top (\mu_E - \mu^{(j)}),$$

where μ_E and $\{\mu^{(j)}\}_{j=1}^i$ are, respectively, the *feature expectations* [AN04] of the expert policy and those of the learned policies up to iteration i . Simulating a one-shot learning scenario, we estimate these quantities from a single system trajectory according to Equation (15.7), i.e.

$$\hat{\mu}(\pi) \triangleq \frac{1}{N} \sum_{n=1}^N \sum_{t=0}^T \gamma^t \phi(\xi^{(n)}(s_t)),$$

for some sufficiently large T , where the state sequence (s_0, s_1, \dots, s_T) is generated under policy π . For more details, we refer to [AN04].

16

Experimental Results

In this chapter, we provide experimental results for two different system types. For the heterogeneous learning scheme (Algorithm 2) used in the policy update step of Algorithm 1, the initial number of exploring agents is set to 50% of the population size and the learning rate is initialized close to 1. Both quantities are controlled by a quadratic decay, which ensures that, at the end of the learning period (i.e., after 200 iterations), the learning rate reaches zero and there are no exploring agents left. Note that these parameters are by no means optimized; yet, in our experiments we could observe that the learning results are largely insensitive to the particular choice of parameter values. Since the agents' observation space is one-dimensional in both experiments, we use a simple tabular representation for the learned Q-function. For higher-dimensional problems, we must resort to function approximation [LP03].

16.1 The Vicsek Model

First, we test our framework on the Vicsek model of self-propelled particles [Vic+95]. The model consists of a fixed number of particles — or agents — living in the unit square $[0, 1] \times [0, 1]$ with periodic boundary conditions. Each agent n moves with a constant absolute velocity v and is characterized by its location $x_t^{(n)}$ and orientation $\theta_t^{(n)}$ in the plane, as summarized by the local state variable $s_t^{(n)} \triangleq (x_t^{(n)}, \theta_t^{(n)})$. The time-varying neighborhood structure of the agents is determined by a fixed interaction radius ρ . At each time instant, the agents' orientations get synchronously updated to the average orientation of their neighbors (including themselves) plus some additive random perturbations $\{\Delta\theta_t^{(n)}\}$, i.e.

$$\begin{aligned}\theta_{t+1}^{(n)} &= \langle \theta_t^{(n)} \rangle_\rho + \Delta\theta_t^{(n)}, \\ x_{t+1}^{(n)} &= x_t^{(n)} + v_t^{(n)}.\end{aligned}\tag{16.1}$$

Herein, $\langle \theta_t^{(n)} \rangle_\rho$ denotes the mean orientation of all agents within the ρ -neighborhood of agent n at time t , and $v_t^{(n)} \triangleq v \cdot [\cos \theta_t^{(n)}, \sin \theta_t^{(n)}]$ is the velocity vector of agent n .

Our goal is to learn a policy model from recorded agent trajectories that reproduces the system behavior in Equation (16.1) using the proposed IRL framework. As a simple observation mechanism, we let the agents in the system compute the angular distance

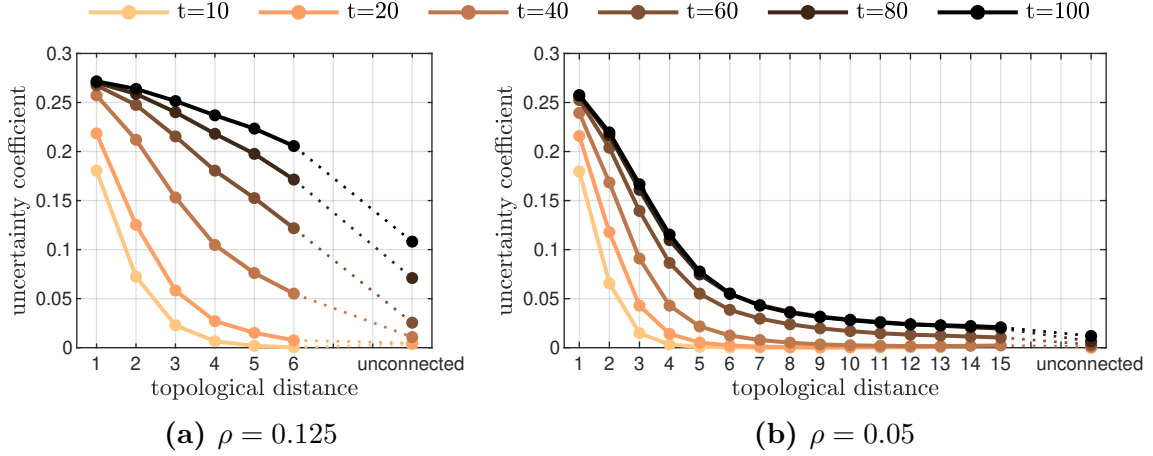


Figure 16.1: Uncertainty coefficient as a function of the topological distance between two agents in the Vicsek system. The curves show the coefficient values at different simulation time points for two interaction radii. The results were obtained from a kernel density estimate of the joint distribution of two agents’ orientations based on 10000 Monte Carlo runs.

to the average orientation of their neighbors, i.e., $o_t^{(n)} = \xi^{(n)}(s_t) \triangleq \langle \theta_t^{(n)} \rangle_\rho - \theta_t^{(n)}$, giving the agents the ability to monitor their local misalignment. For simplicity, we discretize the observation space $[0, 2\pi)$ into 36 equally-sized intervals (visible in Figure 16.2) that build the feature representation ϕ (Section 15.3). Furthermore, we coarse-grain the space of possible direction changes to $[-60^\circ, -50^\circ, \dots, 60^\circ]$, resulting in a total of 13 actions available to the agents. For the experiment, we use a system size of $N = 200$, an interaction radius of $\rho = 0.1$ (if not stated otherwise), an absolute velocity of $v = 0.1$, a discount factor of $\gamma = 0.9$, and a zero-mean Gaussian noise model for $\{\Delta\theta_t^{(n)}\}$ with a standard deviation of 10° . These parameter values are chosen such that the expert system operates in an ordered phase [Vic+95].

16.1.1 Local Coupling and Redundancy

In Section 15.2.1, we claimed that the dependence of any two agents in a swarm system declines with increasing distance between those agents, due to the local coupling structure of the swarm (Chapter 14). In this section, we substantiate our claim by analyzing the coupling strength in the system as a function of the agents’ topological distance. As a measure of (in-)dependence, we employ the *uncertainty coefficient* [Pre+07]—a normalized version of the mutual information shared between two agents—to quantify the amount of information we can predict about one agent’s orientation by observing that of the other. As opposed to Pearson’s correlation coefficient, this quantity is able

to capture nonlinear dependencies, and hence, it is more meaningful in the context of the Vicsek model, whose state dynamics are inherently nonlinear (Equation 16.1).

Figure 16.1 depicts the result of our analysis, which nicely reveals the spatio-temporal flow of information in the system. It confirms that the information exchange between the agents strongly depends on the strength of their coupling, which is determined by (i) their topological distance and (ii) the average number of connecting links (seen from the fact that, for a fixed topological distance, the dependence grows with the interaction radius ρ). We further notice that, for increasing radii, the level of information exchange increases even for agents that are temporarily not connected with each other through the system, due to the higher chance of having been connected at some earlier stage.

16.1.2 Learning Results

A fundamental problem inherent to any IRL approach is the assessment of the extracted reward function, because there is generally no unique solution to the estimation problem (Section 2.2.2). Nonetheless, there are several ways to verify if the estimated reward model is plausible in the context of the observed task.

The simplest way to check the plausibility of the learned model is by subjective inspection: since a system’s reward function can be regarded as a concise description of the performed task, the found estimate should provide a sufficiently intuitive explanation for the observed behavior. As we can see from Figure 16.2, this is clearly the case for the obtained estimation result. Even though there exists no “true” reward model for the Vicsek system, we can tell from the system dynamics in Equation (16.1) that the agents tend to align over time—a behavior that can be induced by providing higher rewards for synchronized states and giving lower (or negative) rewards for local misalignment. Inspecting the induced system dynamics in Figure 16.3, we observe that the learned reward model indeed reproduces the behavior of the expert system, both during the transient phase and at stationarity. Note that the absolute direction of motion is irrelevant since the Vicsek dynamics are invariant to rotations of the coordinate system.

To quantify the accuracy of the behavior reconstruction, we compare both policies in terms of the achieved *order parameter* [Vic+95], which provides a measure for the total alignment of the swarm, i.e.

$$\omega_t \triangleq \frac{1}{Nv} \left| \sum_{n=1}^N v_t^{(n)} \right| \in [0, 1],$$

with values close to 1 indicating strong synchronization of the agents. Figure 16.4 depicts the slope of this parameter for different system policies, including the expert policy and two types of learned policies. From the result, we can see a considerable performance gain for the proposed value estimation scheme (Equation 15.7) compared to the single-agent approach (Equation 15.6), where only the rewards of one agent are considered. This reconfirms our findings from Section 16.1.1, because the increase in performance has to stem from the additional information provided by the remaining agents. As a further reference, we also show the result for a handcrafted reward model, where we provide a non-zero reward only if the local observation of the agent falls into the discretization interval centered around 0° misalignment. As the results reveal, the learned reward model significantly outperforms the ad-hoc solution.

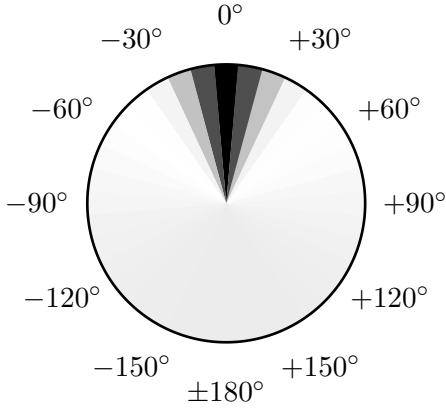


Figure 16.2: Learned reward model for the Vicsek system as a function of the agent's local misalignment. The values are averaged over 100 Monte Carlo runs. Dark color indicates high reward.

16.2 The Ising Model

In the second experiment, we apply the proposed IRL framework to the well-known Ising model [Isi25]. In our case, the system consists of a finite grid of atoms (i.e., agents) of size 100×100 . Each agent has an individual spin $q_t^{(n)} \in \{+1, -1\}$, which, together with its position on the grid, forms its local state, i.e., $s_t^{(n)} \triangleq (x^{(n)}, y^{(n)}, q_t^{(n)})$.

For the experiment, we consider a static (5×5) -neighborhood system, meaning that each agent interacts with its 24 closest neighbors, i.e., agents with a maximum Chebyshev distance of 2. Based on this neighborhood structure, we define the global system energy as

$$E_t \triangleq \sum_{n=1}^N \sum_{m \in \mathcal{N}_n} \mathbf{1}(q_t^{(n)} \neq q_t^{(m)}) = \sum_{n=1}^N E_t^{(n)},$$

where \mathcal{N}_n and $E_t^{(n)}$ are, respectively, the neighborhood and local energy contribution of

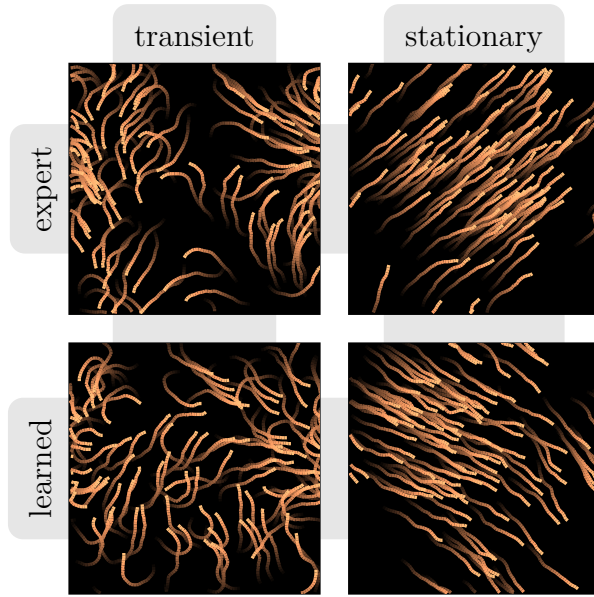


Figure 16.3: Illustrative trajectories of the Vicsek dynamics, generated under the expert policy and a policy learned using the proposed IRL framework. A color-coding scheme is used to indicate the temporal progress. The absolute direction of motion is irrelevant since the Vicsek dynamics are invariant to rotations of the coordinate system.

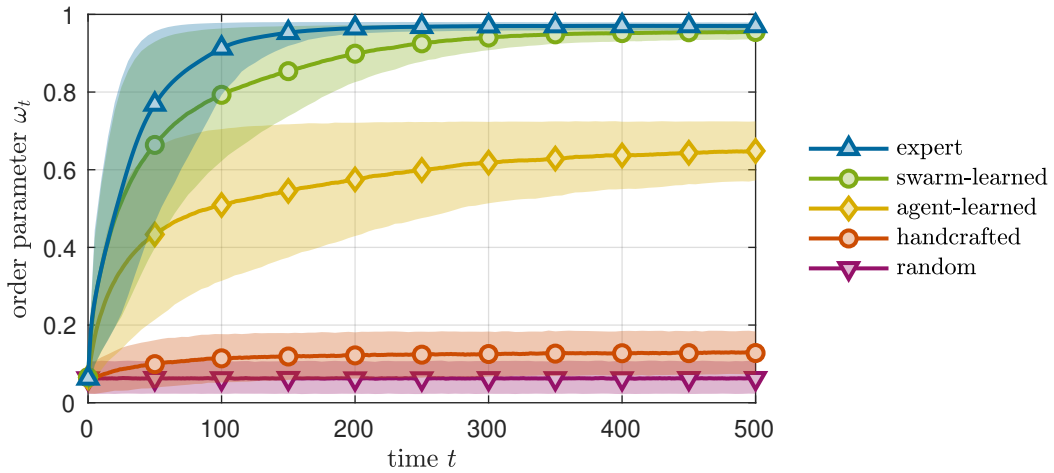


Figure 16.4: Slopes of the order parameter ω_t in the Vicsek system. From top to bottom, the curves show the results for (i) the expert policy, (ii) the learned IRL policy, (iii) the result that is obtained when the feature expectations are estimated based on just one single agent, (iv) for a handcrafted reward function, and (v) for randomly generated policies. For the optimal policy, we show the empirical mean and the corresponding 10% and 90% quantiles, based on 10000 Monte Carlo runs. For the learned policies, we instead show the average over 100 conditional quantiles (since the learning outcome is subject to the stochasticity of the system), each based on 100 Monte Carlo runs with fixed policy.

agent n , and $\mathbf{1}(\cdot)$ denotes the indicator function. Like the order parameter for the Vicsek system (Section 16.1), the global energy E_t serves as a measure for the total alignment of the agents, with an energy value of 0 indicating complete state synchronization. As before, we give the agents the ability to monitor their local misalignment, this time provided in form of their local energy contribution, i.e., $o_t^{(n)} = \xi^{(n)}(s_t) \triangleq E_t^{(n)}$.

In the following, we consider two possible actions for the agents, i.e., *keep the current spin* and *flip the spin*. The system dynamics are chosen such that the executing agent transitions to the desired state with probability 1. With these actions, a meaningful objective for the agents is to reach a global state configuration of minimum energy. As in Section 16.1, we aspire to learn a behavioral model for this task by observing an expert system. In this case, we consider an expert behavior that synchronizes the agents via local majority voting, i.e., using a policy that iteratively assigns each agent to the majority spin in its neighborhood.*

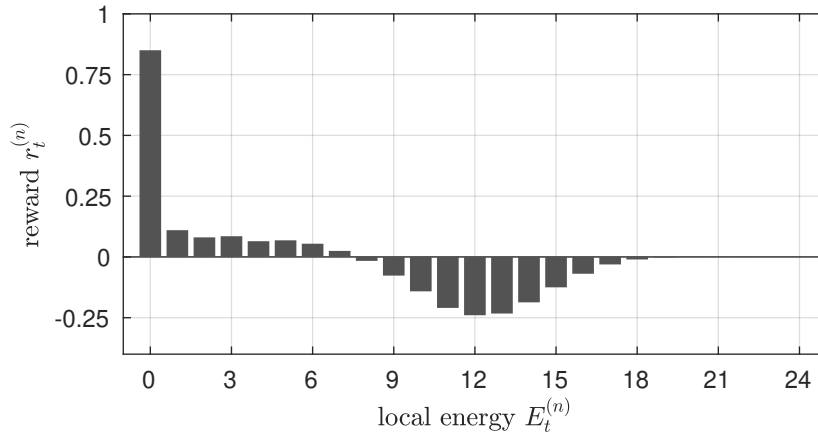


Figure 16.5: Learned reward model for the Ising system as a function of the agent’s local misalignment. The values are averaged over 100 Monte Carlo runs.

Figures 16.5 and 16.6 depict, respectively, the learned mean reward function and the slopes of the global energy for different behavior policies. As in the Vicsek example, the extracted reward function provides an intuitive explanation for the expert system dynamics. Note in particular that assigning a neutral reward to states of high local energy is plausible since a strong local misalignment indicates high synchronization of opposite spin in the agent’s neighborhood. Moreover, using the swarm-based value

* Note that this policy implements a synchronous version of the *iterated conditional modes* algorithm [Bes86], which is guaranteed to translate the system to a state of locally minimum energy.

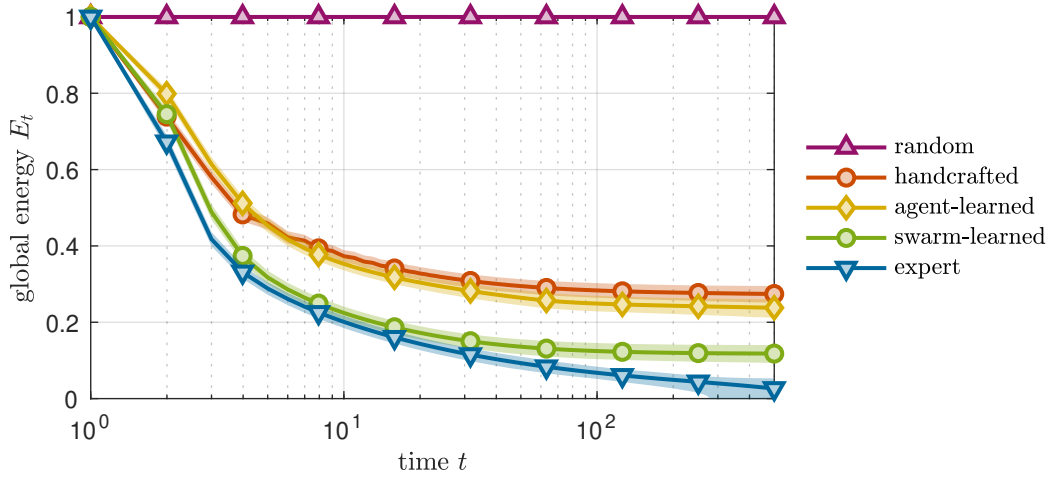


Figure 16.6: Slopes of the global energy E_t in the Ising system. The curves are analogous to those in Figure 16.4, showing the same underlying trend.

estimation approach, we observe the same qualitative performance improvement in terms of the achieved synchronization level as for the Vicsek system, both when compared to the single-agent estimation scheme and to the handcrafted reward model.

17

Summary

Our objective in this part has been to extend the IRL principle to the case of homogeneous multi-agent systems, to obtain an inference framework for learning local reward models that can explain—and reproduce—the emergent behavior of a swarm system. By exploiting the homogeneity property of the introduced swarMDP model, we showed that both value estimation and policy update required for the IRL procedure can be performed based on local experience gathered at the agent level. The so-obtained reward models were used to guide the agents during the proposed heterogeneous learning procedure, yielding local policy models that mimic the observed expert behavior. We tested the framework on two types of system dynamics, covering both static and dynamic agent neighborhoods, where it produced policies that achieved performance levels close to those of the expert systems.

PART IV

Multi-Agent Systems: Continuum Modeling

18 Motivation	140
19 The Agent Model	142
20 The Continuum Model	151
21 Reinforcement Learning in the Continuum Model	156
22 Experimental Results	159
23 Summary	169

In Part III, we turned our attention to multi-agent environments and asked the question what local optimality criterion is being optimized in a given network of agents. To answer the question, we devised an IRL algorithm that allows to estimate the latent reward function of a swarm based on observed trajectory information. An important aspect of this algorithm is the simulation of the network under a given system policy and the subsequent optimization of the agent behavior with respect to the current reward estimate through an RL scheme. Unfortunately, the feasibility of both these procedures crucially depends on the size of the target network—a general problem attributable to the agent-based modeling paradigm. In this part, we focus on the computational aspects of swarm modeling and present a continuum formulation of the model that provides a tractable approximation of the system dynamics for large agent numbers, offering an alternative optimization strategy for RL and IRL in large-scale networks.

Notation While the presented model is compatible with the time-discrete framework in Part III, its formulation is based on a continuous-time version of the underlying system dynamics. To emphasize this difference, we switch to a notation that is more common in the stochastic optimal control literature, which conceptually builds on the theory of stochastic differential equations. In terms of symbols, this change affects

the state variables, for which we use the letter X , and the action variables, which are replaced with continuous-valued control variables denoted by the letter u .

18

Motivation

Agent-based modeling is a powerful and widely-used approach to system design that offers an explicit and intuitive way to describe the individual components of a system and their relationships in a network. Depending on the network characteristics, however, the amount of interactions between the agents can grow rapidly in large networks, which may cause an immense computational overhead when simulating the joint dynamics of the system. Yet, it is precisely the large-scale, *emergent* properties of a network, such as spontaneous order, robustness and rapid information exchange, that are interesting to study from a scientific and engineering point of view [VZ12]. Typical questions that arise in this context are: how is the connectivity/degree of observability in the system related to the feasibility of a given task [Sip99; LB95; CM95]? How does the global system behavior depend on the number of agents involved in the system process [Vic+95; Hay02]? What is the limiting system behavior as the size of the agent population becomes large [Aum64] and, in particular, how can we learn effective control policies for such large-scale regimes?

In this last part of the dissertation, we present an alternative approach to homogeneous system modeling that is, in contrast to classical agent-based paradigms, specifically designed for large-scale decision-making problems. Our model is based on a continuum description of large groups of cooperatively interacting agents, which we derive under the assumption that each agent in the network has access to local information only and is, hence, partially informed about the global system state. A particular focus lies on the interconnection of the derived model with the principle of reinforcement learning, which also offers the possibility to apply the framework in the context of IRL (see Part III). To this end, we discuss and compare the effect of different learning paradigms on the success of the collective tasks, which the agents learn to solve through either local or global reward feedback.

The proposed framework is useful in two ways: i) for multi-agent scenarios, it provides a computational approach to handling large-scale distributed decision-making problems

and learning decentralized control policies. In particular, it can be used to examine the limiting behavior of a swarm system as the number of agents approaches infinity, bypassing finite-size effects inherent to agent-based models and avoiding the computational bottlenecks of large-scale agent simulations. ii) For single-agent systems, it offers an alternative approximation scheme for evaluating expectations of state distributions, which is shown to be advantageous in stochastic control problems with high-variance returns.

Related Work

In the past, different approaches have been proposed that move away from an explicit agent-based system representation to a more macroscopic form of modeling. A basic overview of techniques can be found in the survey paper by Brambilla et al. [Bra+13]. A considerable part of the works mentioned in the paper is based on *rate equation models* (see, for example, [LMG05; MIM99; CM06]), which describe the state transitions of the agents in the system on a global scale, and hence, provide the highest level of abstraction from the microscopic agent-based paradigm toward a macroscopic system description. By completely ignoring the spatial properties of the underlying system, these models can only provide a high-level view on the system dynamics, without considering potentially important local effects caused by the interactions of the agents. Two alternative types of models, which explicitly capture the spatial dynamics of a system, are *continuous spatial automata* [Mac90] and *amorphous media* [Bea05; Abe+00]. Although they have not emerged directly from the classical agent-based paradigm, the underlying methodology shows many parallels to the continuum concept presented in this part. On the side of agent-inspired modeling, there are finally the frameworks of *mean field games* [LL07; Aum64] and *Brownian agents/active particles* [Sch03]. Similar to our approach, these frameworks describe the spatial interactions of the agents at a macroscopic level, using a continuum description of the system dynamics. Moreover, the model in [Sch03] was later extended by Hamann and Wörn [HW08] to explicitly account for communication between the agents. Though these frameworks provide elegant mathematical tools to model the large-scale behavior of a system, existing work mostly focuses on analyzing the collective dynamics of the systems while assuming that the behavior of the individual agent is known.

In this part of the dissertation, the focus is different: instead of assuming fixed agent characteristics, we consider the system process from an engineering perspective and explicitly address the question of how we can optimize the local interactions of the agents in order to stimulate a certain type of macroscopic system behavior. To this

end, we not only use an explicit policy model as a free design parameter, but also demonstrate how that model can be trained via different feedback mechanisms from the system process.

19

The Agent Model

Similar to the discrete-time setting discussed in Part III, we start by considering a finite-size system of N agents. Each agent $n \in \{1, \dots, N\}$ is associated with a *local state*, denoted $X_n(t) \in \mathcal{X} \subseteq \mathbb{R}^d$, which is used to describe the agent's trajectory in our model. Herein, $t \in \mathbb{R}_{\geq 0}$ denotes continuous time and \mathcal{X} is the d -dimensional state space of the system. The collection of all agents' states is referred to as the *global state* of the system, which we represent by the time-dependent state matrix $X(t) \triangleq [X_1(t), X_2(t), \dots, X_N(t)] \in \mathcal{X}^N \subseteq \mathbb{R}^{d \times N}$.

19.1 System Dynamics

In contrast to the swarMDP model in Part III, whose dynamics are characterized in terms of a discrete-time transition model, we describe the interaction of the agents through a set of locally coupled stochastic differential equations. More specifically, we assume that each agent's trajectory follows a controlled Itô diffusion [Kry80], i.e.

$$dX_n(t) = h(X_n(t), u_n(t))dt + dW_n(t). \quad (19.1)$$

Herein, $u_n(t) \in \mathcal{U} \subseteq \mathbb{R}^U$ is the *local control command* executed by agent n at time t , where \mathcal{U} is the set of admissible controls, the function $h : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}^d$ describes the agent's individual dynamics, and $W_n(t) \in \mathbb{R}^d$ represents a noise term, which we model as a Wiener process with independent components for each state dimension. Also, we assume that the noise processes $\{W_n\}_{n=1}^N$ are pairwise independent, which implies that

$$\mathbb{E}[W_{n,i}(t)W_{m,j}(t')] = 2D\delta_{i,j}\delta_{n,m}\min(t, t'),$$

where δ denotes Kronecker's delta, $W_{n,i}(t)$ refers to the i th component of the noise process affecting agent n at time t , and $D \in \mathbb{R}_{\geq 0}$ is a scalar constant referred to as the diffusion coefficient.

While the stochasticity of the swarMDP system is captured implicitly in the transition model \mathcal{T} , the additive term $W_n(t)$ makes the randomness of the state transitions in Equation (19.1) explicit. The role of this term is twofold: on the one hand, it allows to model systems whose state transitions are inherently random (i.e., systems with truly stochastic state dynamics); on the other hand, it can be used to summarize the effect of unmodeled system parameters (e.g., a change of an agent's state due to external forces). In both cases, the underlying assumption is that the resulting state increments can be described by a Gaussian distribution [Bil99]. Note, however, that this assumption does not carry over to the state variable X_n itself, which is generally non-Gaussian due to the influence of the dynamics h and the control term $u_n(t)$.

19.2 Observation Model

The extent to which agent n adjusts its local state $X_n(t)$ via $u_n(t)$ (Equation 19.1) depends on the states of the other agents or, more precisely, on the agent's k -dimensional *local observation* $Y_n(t) \in \mathcal{Y} \subseteq \mathbb{R}^k$ of the system state $X(t)$. Here, the symbol \mathcal{Y} denotes the observation space of the agents. Formally, we describe these agent-related quantities through an *observation model* $\xi : \mathcal{X} \times \mathcal{X}^N \rightarrow \mathcal{Y}$, which specifies how the global agent configuration is perceived from any state in \mathcal{X} (compare definition in Chapter 14). More specifically, $\xi(x, X(t))$ tells us how the system state $X(t)$ is perceived from a given agent location x . Consequently, ξ relates $X(t)$ to the local agent observations $\{Y_n(t)\}$, e.g., agent n 's local observation $Y_n(t)$ can be accessed as

$$Y_n(t) = \xi(X_n(t), X(t)).$$

As for the swarMDP model, a generalization to stochastic observation models is possible but not considered (compare discussion on stochastic controllers in Remark 5). The reason why we adopt a two-argument notation $\xi(x, X(t))$ with an independent observer state x instead of resorting to explicit agent-based indexing as in Part III will become clear when we arrive at the continuum formulation in Chapter 20.

Since the agents in the swarm are assumed to be identical and thus interchangeable (Chapter 14), we allow only symmetric observation models that ignore the ordering of the agents and respect their interchangeability. More specifically, we assume the observation model to be of the form

$$\xi(x, X(t)) \triangleq \frac{1}{|\mathcal{N}_{\langle x \rangle}|} \sum_{n \in \mathcal{N}_{\langle x \rangle}} g(x, X_n(t)), \quad (19.2)$$

where g is the *interaction potential* of the system (explained below) and $\mathcal{N}_{\langle x \rangle}$ denotes the set of all agents in some *spatial neighborhood* $\mathcal{B}(x) \subseteq \mathcal{X}$ of x , i.e.

$$\mathcal{N}_{\langle x \rangle} \triangleq \{n : X_n(t) \in \mathcal{B}(x)\}.$$

Note that we keep the time-dependence of $\mathcal{N}_{\langle x \rangle}$ implicit to simplify our notation. To access the *agent-neighborhood* of a specific agent n , we further introduce the shorthand notation $\mathcal{N}_n \triangleq \mathcal{N}_{\langle X_n(t) \rangle}$ and write N_n to denote its cardinality, i.e., $N_n \triangleq |\mathcal{N}_n|$.

The set $\mathcal{B}(x)$ can be thought of as the “observable region” of an agent located at x , i.e., the set of observable other agents’ states. Within that region, the vector-valued function $g : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Y}$ describes the partial contribution of a specific agent to another agent’s local percept of the system state. More specifically, $g(x, y)$ tells us how an individual agent at state y is perceived from state x or, equivalently, how an agent at x is influenced by an agent located at y . We will see various examples of this relationship in Chapter 22.

Because the agents receive their observations according to their local neighborhood (Equation 19.2), we naturally assume that each agent n knows its neighborhood size N_n . Note that the summation in the equation arises from the inherent homogeneity assumption of our model: not only are the agents homogeneous in their architecture, they also treat other agents in their neighborhood interchangeably (i.e., agent m located at state x has the same effect on agent n as agent k located at x). This interchangeability of agents is a necessary requirement for our continuum model in Chapter 20, where each agent will be eventually surrounded by an infinite number of other agents so that a discrimination between neighbors becomes impossible.

Remark 2 (Partial observability). Partial observability is a natural characteristic of virtually all types of distributed systems, and hence, it is an integral part of our system model. Notice that partial observability can manifest itself in two different ways, which are both explicitly reflected in the proposed observation model: i) the limitation of the agents’ observation range, described through the neighborhood set \mathcal{B} ; ii) the restricted information content of the observations within that neighborhood, as prescribed by the interaction potential g . The former is typically a direct consequence of the agents’ (physically) limited observation capabilities, while the latter is determined by the specific type(s) of sensing modality used by the agents.

Remark 3 (Indicator function). Notice that Equation (19.2) can be rewritten in a more explicit form, which will be used for the derivation of the continuum model in

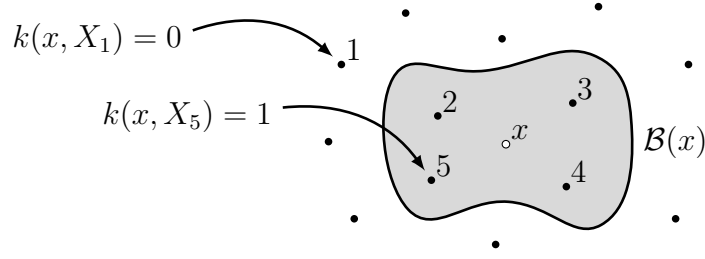


Figure 19.1: Schematic illustration of the neighborhood scheme. The spatial neighborhood $\mathcal{B}(x)$ of state x is highlighted in gray. In that region, the neighborhood indicator function k returns the value 1; outside, the function value is 0. Accordingly, the agent-neighborhood at state x is $\mathcal{N}_{\langle x \rangle} = \{2, 3, 4, 5\}$.

Chapter 20, i.e.

$$\xi(x, X(t)) = \frac{\sum_{n=1}^N g(x, X_n(t)) k(x, X_n(t))}{\sum_{m=1}^N k(x, X_m(t))}. \quad (19.3)$$

Herein, the function $k : \mathcal{X} \times \mathcal{X} \rightarrow \{0, 1\}$ indicates whether an agent at some state y is in the spatial neighborhood of another state x , i.e.

$$k(x, y) \triangleq \begin{cases} 1 & \text{if } y \in \mathcal{B}(x), \\ 0 & \text{otherwise.} \end{cases}$$

An illustration of this neighborhood scheme is provided in Figure 19.1.

Remark 4 (Environmental information). In Equation (19.2), the local observations $\{Y_n(t)\}$ have been defined in terms of pairwise interactions between the agents, as indicated by the explicit summation over neighbors. One possibility to consider also environmental information (i.e., information from the surrounding of an agent that is independent of the other agents) is to extend the observation model by an additional number of dimensions that only take the agent's own state as an argument. However, we can stick with the more compact form in Equation (19.2) without loss of generality if we assume that some dimensions of g implicitly depend only on the first argument passed to the function. Examples of this implicit scheme are given in Equations (22.2) and (22.3).

19.3 System Controller

Having defined the system dynamics and the observation model, we may now focus on the control signals $\{u_n(t)\}$ and describe the interactions between the agents. In order to preserve the homogeneity of our system (Chapter 14), we specify the coupling

mechanism in the form of a *system-wide control policy* $\pi_\theta : \mathcal{Y} \rightarrow \mathcal{U}$, i.e.

$$u_n(t) = \pi_\theta(Y_n(t)), \quad (19.4)$$

which translates the agents' observations $\{Y_n(t)\}$ into local control commands $\{u_n(t)\}$. The policy is parametrized by a *control parameter* $\theta \in \Theta \subseteq \mathbb{R}^C$ with corresponding parameter space Θ . Note that Θ implicitly defines the control set \mathcal{U} through the functional form of π_θ .

Putting all ingredients together, we can write Equation (19.1) in the more explicit form

$$\begin{aligned} dX_n &= h\left(X_n, \pi_\theta(\xi(X_n, X))\right)dt + dW_n \\ &= h\left(X_n, \pi_\theta\left(\frac{1}{N_n} \sum_{m \in \mathcal{N}_n} g(X_n, X_m)\right)\right)dt + dW_n. \end{aligned} \quad (19.5)$$

In Chapter 21, we will consider the system from a reinforcement learning perspective and our goal will be to find a particular θ such that the resulting network optimally solves a given task. For now, however, we may assume that θ is fixed.

Remark 5 (State exploration). By restricting our interaction model to the class of *deterministic reactive policies* (i.e., where $u_n(t)$ is a function of the current observation $Y_n(t)$ only), we focus on settings with agents that do *not* explicitly face the problem of information gathering—see *Discussion and Outlook*. In particular, inspired by the characteristics of natural swarm systems, we assume a simplistic system architecture that bypasses the requirement of long-term agent memory by replacing exploration strategies executed at the agent level with collective strategies deployed at the swarm level.* As shown by the examples in Chapter 22, this architecture is indeed flexible enough to solve a number of common swarm-related tasks. However, it should be mentioned that there are situations for which an extension to stochastic system controllers may be necessary, for example, to model the effect of spontaneous symmetry breaking in natural swarms. For the special case of linear dynamics h and Gaussian control policies, such an extension is straightforward since the stochastic component of the controller can be absorbed into the diffusion coefficient D of the noise processes $\{W_n\}$.

Remark 6 (Information processing). In Equation (19.5), π_θ operates directly on the raw (cumulative) sensory information gathered by the agents. The use of more complex

* Note that both these exploration types are again different from the exploration in policy space, which we discuss in detail in Chapter 21.

observational features to control the agent behavior can be easily realized with the help of an additional preprocessing stage that first extracts the required high-level information from the agent's sensory input before passing it to the controller. However, both these formulations are mathematically equivalent since any preprocessing stage can be absorbed into π_θ ; hence, we may stick with the more compact form in Equation (19.5) without loss of generality. Nevertheless, for the overall picture of the information processing chain, it can be helpful to keep that preprocessing step in mind. A simple example is discussed in Section 22.2 (Equation 22.1).

19.4 Reward Models

With regard to the forthcoming policy optimization procedure described in Chapter 21, we finally need to define a learning objective for our system. As in Part III, we define that objective via a scalar *reward signal* $r(t)$, which serves as a feedback from the system process that allows to assess the system's instantaneous state in the context of the collective task. In the following, we consider the *finite-horizon value criterion*, which measures the total reward accumulated over a certain period of time T , i.e.

$$V_\theta \triangleq \mathbb{E} \left[\int_0^T r(t) dt \mid \pi_\theta \right]. \quad (19.6)$$

Herein, the expectation is with respect to the underlying random system trajectory generated under π_θ , where we assume that the initial system state is drawn from some fixed state distribution.

While the reward mechanism has been defined only *locally* in Part III, we now consider a more general setting where we allow arbitrary feedback from the system process to train our network. To this end, the following two types of reward mechanisms are considered.

19.4.1 Global Reward Model

From a macroscopic point of view, a natural approach to assess the behavior of the network is to use a reward mechanism that directly evaluates the global system state, i.e.

$$r^G(t) \triangleq R^G(X(t)), \quad (19.7)$$

where $R^G : \mathcal{X}^N \rightarrow \mathbb{R}$ is a score function assessing the entire agent configuration. For example, in a positioning task, R^G could be the (negative) sum of the agents' distances

to a target location or, alternatively, in an aggregation task, it could be the (negative) average pairwise distance between the agents.

Considered from a system optimization perspective, the global reward signal $r^G(t)$ can be interpreted as a direct feedback from the system state $X(t)$ to a centralized critic system that monitors the global system state and utilizes the provided reward information to optimize the agent behavior through π_θ . In fact, if we treat our system model as a black box and ignore its internal decentralized structure, the setting can be interpreted as a single-agent scenario with the central critic system as the only decision-maker.

19.4.2 Local Reward Model

The computation of the global reward signal as defined in Equation (19.7) requires complete knowledge of the global system state $X(t)$, which, in a distributed network with many agents, may not be available at any point in the system. To account for this information loss, we also consider the decentralized setting from Part III, where each agent n is provided its own local reward signal $r_n(t)$ instead, i.e.

$$r_n(t) \triangleq R^L(Y_n(t)) = R^L(\xi(X_n(t), X(t))). \quad (19.8)$$

Herein, $R^L : \mathcal{Y} \rightarrow \mathbb{R}$ is a score function used by the agents to assess their local state configuration captured in the form of the local observations $\{Y_n(t)\}$. Assuming that the scalar reward signals $\{r_n(t)\}$ can be accessed by a central critic system, we can define our performance measure indirectly via the local agent rewards, i.e.

$$r^L(t) \triangleq \frac{1}{N} \sum_{n=1}^N r_n(t). \quad (19.9)$$

From a learning perspective, we can think of this scenario as a setting where each agent reports its local reward—in this context to be interpreted as a summary description of its local state configuration—to a centralized learning system, whose goal is to coordinate the system behavior based on the received local feedback.

Remark 7 (Global critic, local actors). For both described reward mechanisms, learning takes place centrally at the critic system, irrespective of whether the behavior is evaluated globally or locally. Yet, it is important to note that, once the learning phase is completed, the agents no longer rely on any centralized information, and hence, they can act autonomously without further instructions from the critic. The learning architecture thus resembles an actor-critic network [Gro+12] in which the global critic

system serves as a fusion center during the training period but where the trained actor (i.e., the agent’s policy) relies only on local information. A particular learning strategy that exploits this special structure can be found in [HŠN17]. Also, note that extensions of the proposed information processing pipeline are conceivable that do not involve a fusion center at all (see *Discussion and Outlook*).

19.5 Value Estimation

In order to assess the performance of a particular system policy in the context of a given task, we need to estimate the corresponding *value* (Section 19.4). Writing Equation (19.6) in a more explicit form, we obtain

$$\begin{aligned} V_\theta &= \int_{\Omega} \int_0^T R^G(X(t; \chi)) \, dt \, P(d\chi) \\ &= \int_{\mathcal{X}^N} \int_0^T R^G(x) p(x, t) \, dt \, dx, \end{aligned} \quad (19.10)$$

where Ω and P denote, respectively, the underlying sample space and probability measure of the state process X , and $p(\cdot, t)$ is the corresponding marginal density at time t .*

The notation $X(\cdot; \chi)$ refers to a particular sample path of the system, i.e., a specific system trajectory generated under the given system dynamics. To keep the notation simple, we omit the system policy π_θ in the above equations and implicitly assume that X is controlled by π_θ according to Equation (19.5).

Having the form of an expectation, the outer integral in Equation (19.10) can be approximated via Monte Carlo integration, i.e.

$$\hat{V}_\theta^{\{S\}} = \frac{1}{S} \sum_{s=1}^S \int_0^T R^G(X(t; \chi^{\{s\}})) \, dt, \quad (19.11)$$

where $\chi^{\{s\}} \sim P$, and S is a specified number of system rollouts. To evaluate this expression for a particular target network, the remaining integral can be replaced by a discrete-time approximation, which is explained in Chapter 21.

* Note that the value definition in Equation (19.10) is based on the *global* reward model described in Section 19.4.1. However, we can easily switch to a *local* value computation by choosing R^G as the average of the agent-based rewards, i.e., by setting $R^G(X(t)) \triangleq \frac{1}{N} \sum_{n=1}^N R^L(\xi(X_n(t), X(t)))$, which would correspond to the average *private value* introduced in Section 15.1.1.

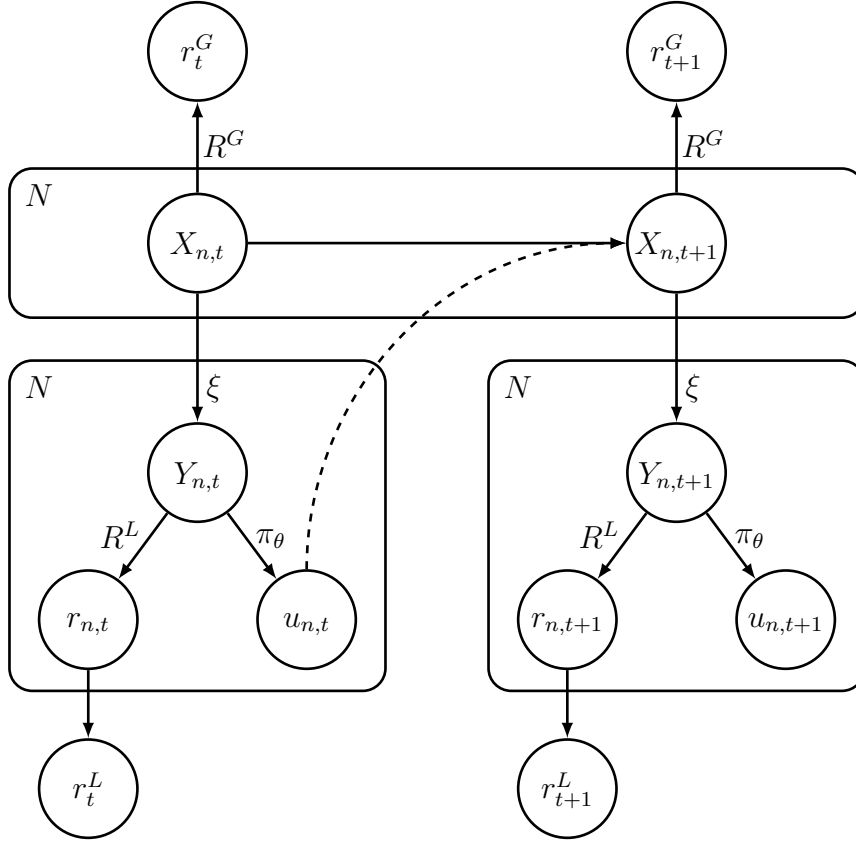


Figure 19.2: The continuous-time N -agent system as a discrete-time swarMDP whose transition dynamics factorize over agents. The local observation $Y_{n,t}$ of agent n at time t is determined by the observation model ξ , based on the local states $\{X_{m,t} : m \in \mathcal{N}_n\}$ of other agents in the neighborhood. The agent transitions to a new state $X_{n,t+1}$ by executing a control command $u_{n,t}$, dictated by the policy π_θ . Note: the dashed arrow indicates dependencies between variables belonging to the same agent. At each time instant, the local and global rewards r_t^L and r_t^G are determined via the functions R^L and R^G according to the agents' local observations and the global system state, respectively.

19.6 The Agent Model in the swarMDP Context

As already indicated at the beginning of this part, the presented agent-based system can be interpreted as a continuous-time version of the swarMDP network from Chapter 14 under certain modeling constraints. More specifically, the relationship can be established for the special case where the global transition dynamics \mathcal{T} of the swarMDP factorize over agents. This means that the agents may still be coupled through their collective choice of actions but, given a particular joint action assignment, the future state of each agent is conditionally independent of all other agents. The special dependency structure of this model is depicted in Figure 19.2, which displays the relationships between all model components and explicitly shows the dynamics factorization (compare

Figure 14.1). Note that we indicate time dependence using subscript notation in the figure, to emphasize the discrete-time nature of the model.

20

The Continuum Model

While the system model in Equation (19.5) is valid for arbitrary network sizes, the computational complexity of the agent-based model can grow up to quadratically with the number of agents in the system, due to the involved pairwise interactions. For large network sizes, this can mean a prohibitively high computational load, especially when the system has to be simulated repeatedly as in the case of IRL (Part III). In this chapter, we present an alternative model description, based on a continuum formulation of the global system state, that allows to approximate the network dynamics in large-scale regimes.

20.1 The Continuum Equation

To describe the system behavior for large agent numbers, it is convenient to switch to a more compact state representation that avoids enumerating each agent individually. The reason is that, as $N \rightarrow \infty$, the individual contribution of a specific agent to the global system dynamics becomes negligible; hence, a continuum-based system model, which describes the dynamics of the collective on a macroscopic scale, offers a more efficient representation of the system's state.

Since in a continuum of agents there is no meaning to the notion of an individual agent, we will drop the concept of using agent-specific quantities — such as states $\{X_n(t)\}$ and observations $\{Y_n(t)\}$ — to describe our system and replace all objects with their continuum-based counterparts. To arrive at the corresponding system model, we consider the global N -agent density* as introduced by Dean [Dea96], i.e.

$$\rho^{(N)}(x, t) \triangleq \frac{1}{N} \sum_{n=1}^N \rho_n(x, t), \quad (20.1)$$

* The N -agent density is not to be confused with the probability density function of a single agent's state (see Section 20.4), which, in contrast to the object defined here, is a deterministic quantity.

which is defined in terms of the following single-agent density functions,

$$\rho_n(x, t) \triangleq \delta(x - X_n(t)). \quad (20.2)$$

Note that the state matrix $X(t)$ and the density $\rho^{(N)}(x, t)$ can be considered as equivalent descriptions of the system state since we generally ignore the ordering of the agents.

In the limit as $N \rightarrow \infty$, the temporal evolution of this quantity, which we in this context refer to as the *continuum density*, is described by the following convection-diffusion equation,

$$\frac{\partial \rho(x, t)}{\partial t} = -\nabla \cdot \left[\rho(x, t) h(x, \bar{u}(x, t)) \right] + D \nabla^2 \rho(x, t), \quad (20.3)$$

where the first component on the right-hand side replaces the drift term in Equation (19.5) and the second component models the cumulative effect of the agent-dependent noise processes. In this equation, the newly introduced *control field*

$$\bar{u}(x, t) \triangleq \pi_\theta(\bar{y}(x, t))$$

and the underlying *observation field*

$$\bar{y}(x, t) \triangleq \frac{\int_{\mathcal{X}} \rho(y, t) g(x, y) k(x, y) dy}{\int_{\mathcal{X}} \rho(y', t) k(x, y') dy'}$$

take over the roles of the agent-specific control and observation signals $\{u_n(t)\}$ and $\{Y_n(t)\}$, respectively.

A detailed derivation of Equation (20.3) is provided in Appendix C. Illustrations of the N -agent density and the continuum density are provided in Figure 22.1, based on the Kuramoto model described in Chapter 22.

20.2 Reward Models

To complete the model, we finally transfer the reward mechanisms from Section 19.4 to the continuum domain. This allow us to evaluate the performance of a particular system policy π_θ using a continuum-based simulation of the network.

20.2.1 Global Reward Model

The global reward mechanism of the continuum model can be defined in a similar way as for the agent-based model, with the subtle difference that the reward is no

longer a function of the state matrix $X(t)$ but becomes a functional of the continuum density $\rho(x, t)$, i.e.

$$r^G(t) \triangleq R^G(\rho(\cdot, t)), \quad (20.4)$$

where $R^G : \mathcal{M} \rightarrow \mathbb{R}$, and \mathcal{M} is the set of density functions on \mathcal{X} , i.e.

$$\mathcal{M} \triangleq \left\{ f : f(x) \geq 0 \quad \forall x \in \mathcal{X}, \int_{\mathcal{X}} f(x) dx = 1 \right\}.$$

While the mathematical forms of the continuum-based and the agent-based reward models differ, both models are conceptually equivalent because $\rho(x, t)$ takes over the role of the state matrix $X(t)$ as $N \rightarrow \infty$. In that sense, $r^G(t)$ remains a performance measure defined on the global system state.

20.2.2 Local Reward Model

For the decentralized setting, we derive the reward mechanism directly from the agent-based model. Starting from Equations (19.8) and (19.9), we write the fused reward signal as

$$r^L(t) = \frac{1}{N} \sum_{n=1}^N R^L(\xi(X_n(t), X(t))).$$

As in the derivation of the continuum equation (Appendix C), we express this quantity using the N -agent density $\rho^{(N)}(x, t)$, which yields

$$r^L(t) = \int_{\mathcal{X}} \rho^{(N)}(x, t) R^L(\xi(x, X(t))) dx.$$

The expression can be simplified with the help of the observation field $\bar{y}^{(N)}(x, t)$, i.e.

$$r^L(t) = \int_{\mathcal{X}} \rho^{(N)}(x, t) R^L(\bar{y}^{(N)}(x, t)) dx.$$

Next, we introduce a corresponding *reward field* $\bar{r}^{(N)}(x, t)$, which provides the reward of a (hypothetical) agent coupled to the system at state x , i.e.

$$\bar{r}^{(N)}(x, t) \triangleq R^L(\bar{y}^{(N)}(x, t)).$$

With this definition at hand, we finally obtain the aggregated reward signal as

$$r^L(t) = \int_{\mathcal{X}} \rho(x, t) \bar{r}(x, t) dx, \quad (20.5)$$

where we have replaced all finite-size quantities with their continuum equivalents.

20.3 Value Estimation

In Section 19.5, we saw that the value of a policy in the agent-based system can be estimated via Monte Carlo integration, based on a finite number of independent system rollouts. For the continuum model, whose system behavior is uniquely captured by the deterministic continuum density $\rho(x, t)$, the estimation scheme is fundamentally different. Here, it is sufficient to consider a single rollout per initial system state, i.e.

$$V_{\theta}^{\rho} = \int_0^T R^G(\rho(\cdot, t)) dt. \quad (20.6)$$

In contrast to the Monte Carlo estimator in Equation (19.11), the estimation accuracy is hence not limited by the number of experiments but by the accuracy of the representation used for the continuum density (see *Discussion and Outlook*).

20.4 Continuum Modeling versus Agent-Based Modeling

Having introduced two alternative views on the learning problem, we conclude the previous sections by highlighting some of the main differences between the two modeling approaches and pointing out some important implications.

Single-Agent Systems and Decoupled System Dynamics

So far, we have considered the continuum framework under the implicit assumption that the agents in the system interact cooperatively with each other in a joint state space. Interestingly, a continuum formulation is not only useful for modeling cooperative decision-making problems but also for single-agent systems or — equivalently — in the special case of a completely decoupled network where no interaction occurs. In such a decoupled system, the simulation of the N -agent state $X(t)$ corresponds to simulating N independent rollouts of the unique underlying single-agent dynamics. Hence, in this context, the continuum density becomes equivalent to the probability density function of the single agent's state (Equation 19.10). To be precise, it follows that

$$\rho(x, t) = p(x, t), \quad (20.7)$$

and the continuum equation in (20.3) reduces to the Fokker-Planck equation [Ris96] that describes the state evolution of the agent. This equivalence holds because, mathematically, there is no difference in modeling the probability of state occurrence of a single agent or modeling the concentration profile of infinitely many identical decoupled agents.

Changing our viewpoint, however, allows us to switch from the sample-based estimator in Equation (19.11) to the continuum computation in Equation (20.6) which, by modeling the agent distribution in Equation (20.7) directly, realizes an average over *all* sample paths of the system. More precisely, for $S \rightarrow \infty$ it holds that

$$\hat{V}_\theta^{\{S\}} \xrightarrow{a.s.} V_\theta^\rho,$$

by the law of large numbers. Due to the virtually infinite sample size, a continuum-based system model can be advantageous in scenarios with high-variance returns as, for example, in problems with sparse or fragmented reward structures (see Section 22.3). This argument holds not only true for the system value but for the computation of arbitrary expectations with respect to the agent's state distribution.

Cooperative Multi-Agent Systems

In the more complex scenario of coupled agent dynamics, the continuum model offers an additional benefit over the agent-based approach. Driven by the local interactions of the agents, the global network dynamics depend, in general, on the number of decision-makers involved in the system process [VZ12]. As $N \rightarrow \infty$, the continuum approach not only provides a more efficient approximation of the system behavior by avoiding the explicit simulation of these interactions (see beginning of this chapter) but also a more accurate one in the sense that it bypasses finite-size effects that would arise in any agent-based simulation. Hence, the continuum formulation offers an elegant alternative to classical agent-based computation for both, single-agent and cooperative systems. However, being a macroscopic system model, the continuum model cannot be used to describe the erratic behavior of an individual agent in the network at the microscopic scale. Consequently, for small network sizes, the predicted system behavior may differ significantly from the true dynamics [HV15; OSK08].

Computational Cost

From a computational point of view, the two modeling approaches have complementary strengths and weaknesses: the feasibility of conducting an agent-based simulation largely depends on the size of the network (see previous paragraph), while for the continuum framework the computational limits are set by the dimensionality of the system state space, which dictates the complexity of the continuum representation. A promising future research direction is, therefore, to consider hybrid approaches that combine the merits of both methods (see *Discussion and Outlook*).

21

Reinforcement Learning in the Continuum Model

In this chapter, we discuss a simple reinforcement learning strategy to optimize the behavior of our system. Herein, we restrict ourselves to policy gradient methods [DNP13], which provide a model-free learning paradigm to perform the system optimization directly in the parameter space of the policy.

While our model is defined in terms of (stochastic) differential equations and hence naturally operates in continuous time, we consider the learning problem on a discretized time scale, which allows us to conduct the optimization using numerical simulations of the network dynamics. To this end, we introduce the random variable \mathcal{R} , referred to as the sample return, which measures the (local or global) reward signal $r(t)$ at regular time intervals $\Delta \triangleq \frac{T}{M}$ for some $M \in \mathbb{N}$, i.e.

$$\mathcal{R} \triangleq \sum_{k=0}^M r(k\Delta). \quad (21.1)$$

Note that the distribution of \mathcal{R} , which we denote by $p_{\mathcal{R}}(\mathcal{R} | \theta)$, depends on the control parameter θ due to the inherent dependence of $r(t)$ on the system policy. Based on the sample return, we define the following objective function,

$$J_{\theta} \triangleq \int_{-\infty}^{\infty} p_{\mathcal{R}}(\tau | \theta) \tau \, d\tau. \quad (21.2)$$

Up to a rescaling with Δ , J_{θ} can be regarded as a discrete-time approximation of the continuous-time value in Equation (19.6) that may, for small enough Δ , be used as a substitute. However, a direct optimization of J_{θ} is unfavorable because—under the assumption of a deterministic system policy π_{θ} (Section 19.3)—the policy update is based on a gradient computation that involves the system’s transition model, which destroys the black box character of the learning mechanism. While a stochastic policy model would generally solve the problem, it artificially injects noise into the system, which increases the variance of the gradient estimator [Seh+10]. The effect is further amplified by the discrete-time simulation of the model, since the number of random decisions fed into the

system process increases with the temporal resolution $\frac{1}{\Delta}$. In the extreme case, the gradient estimate can be degraded up to a point where learning becomes impossible [Mun06].

A principled approach to handling this dilemma is offered by the parameter exploring policy gradient (PEPG) [Seh+10]. The algorithm makes use of an explicit upper-level *exploration policy* $p(\theta | \omega)$, which is defined on the control parameter space Θ itself, parametrized by an additional exploration parameter ω . Accordingly, the objective function in Equation (21.2) is expanded to an upper-level criterion, i.e.

$$\mathcal{J}_\omega \triangleq \int_{\Theta} p(\theta | \omega) J_\theta \, d\theta,$$

which measures the expected system performance for a given parameter value ω . To find an optimal ω , we apply a normalized gradient ascent procedure, i.e.

$$\omega \leftarrow \omega + \alpha z^{-1} \frac{\partial \mathcal{J}_\omega}{\partial \omega},$$

with $z \triangleq \|\frac{\partial \mathcal{J}_\omega}{\partial \omega}\|_2$ and some fixed learning rate $\alpha \in (0, \infty)$. Herein, the gradient is given by

$$\begin{aligned} \frac{\partial \mathcal{J}_\omega}{\partial \omega} &= \int_{\Theta} p(\theta | \omega) \frac{\partial}{\partial \omega} \log p(\theta | \omega) J_\theta \, d\theta \\ &= \mathbb{E} \left[\frac{\partial}{\partial \omega} \log p(\theta | \omega) J_\theta \right]. \end{aligned} \quad (21.3)$$

The above expectation can be estimated using Monte Carlo integration, i.e.

$$\frac{\partial \mathcal{J}_\omega}{\partial \omega} \approx \frac{1}{Q} \sum_{q=1}^Q \frac{\partial}{\partial \omega} \log p(\theta^{\{q\}} | \omega) [\mathcal{R}^{\{q\}} - b].$$

To construct this estimate, we first sample the control parameters, i.e., $\theta^{\{q\}} \sim p(\theta | \omega)$, and keep them fixed while rolling out the system, i.e., $\mathcal{R}^{\{q,s\}} \sim p_{\mathcal{R}}(\tau | \theta^{\{q\}})$. An estimate of the $J_{\theta^{\{q\}}}$ is then obtained as $\mathcal{R}^{\{q\}} \triangleq \frac{1}{S} \sum_{s=1}^S \mathcal{R}^{\{q,s\}}$, from which we subtract the baseline $b \triangleq \frac{1}{Q} \sum_{q=1}^Q \mathcal{R}^{\{q\}}$ to reduce the variance of the estimator [DNP13]. Based on this two-step approach, the policy optimization is effectively shifted from the lower to the upper level, which decouples the exploration process from the system process and avoids the gradient degeneracy problem mentioned earlier.

Once an optimal value ω^* is found, the final low-level parameter θ^* can be set, for example, to the corresponding mean value, i.e.

$$\theta^* \triangleq \int_{\Theta} p(\theta | \omega^*) \theta \, d\theta.$$

Algorithm 3: PARAMETER EXPLORING POLICY GRADIENT

Input: return-generating process $p_{\mathcal{R}}(\tau | \theta)$, parameters α, Q, S, σ^2

Output: control parameter θ

```

1: initialize exploration parameters  $\omega$  randomly
   while stopping criterion not fulfilled
       for  $q = 1, 2, \dots, Q$ 
2:         sample control parameter:  $\theta^{\{q\}} \sim p(\theta | \omega) = \prod_{c=1}^C \mathcal{N}(\theta_c | \omega_c, \sigma^2)$ 
           for  $s = 1, 2, \dots, S$ 
3:             sample return value from the process:  $\mathcal{R}^{\{q,s\}} \sim p_{\mathcal{R}}(\tau | \theta^{\{q\}})$ 
           end for
4:         estimate value for  $\theta^{\{q\}}$ :  $\mathcal{R}^{\{q\}} \leftarrow \frac{1}{S} \sum_{s=1}^S \mathcal{R}^{\{q,s\}}$ 
       end for
5:       compute baseline:  $b \leftarrow \frac{1}{Q} \sum_{q=1}^Q \mathcal{R}^{\{q\}}$ 
6:       estimate policy gradient:  $\delta\omega \leftarrow \frac{1}{Q\sigma^2} \sum_{q=1}^Q (\theta^{\{q\}} - \omega)(\mathcal{R}^{\{q\}} - b)$ 
7:       update exploration parameters:  $\omega \leftarrow \omega + \alpha \cdot \frac{\delta\omega}{\|\delta\omega\|_2}$ 
   end while
8: return  $\theta \leftarrow \omega$ 
    
```

For our experiments in the subsequent sections, we use a factored Gaussian exploration policy, where the exploration parameters $\{\omega_c\}$ are used to parametrize the means of the respective low-level control parameters $\{\theta_c\}$, i.e.

$$p(\theta | \omega) = \prod_{c=1}^C \mathcal{N}(\theta_c | \omega_c, \sigma^2).$$

For this class of exploration policy, the gradient of the log density in Equation (21.3) takes the form

$$\frac{\partial}{\partial \omega} \log p(\theta | \omega) = \frac{\theta - \omega}{\sigma^2},$$

which results in the learning scheme outlined in Algorithm 3.

22

Experimental Results

To demonstrate the working principle of the proposed continuum framework, we consider a system of interacting agents on a circle. The system is based on the well-known model by Kuramoto [Kur75] (Figure 22.1), which has been used extensively in the past to study the synchronization behavior of groups of coupled oscillators. The basic Kuramoto dynamics are characterized by the following stochastic differential equation,

$$dX_n(t) = \left[\varphi_n + \frac{K}{N} \sum_{m=1}^N \sin(X_m(t) - X_n(t)) \right] dt + dW_n(t),$$

where φ_n is the natural frequency of oscillator n , and K represents the coupling strength of the system.

Comparing the system model with the one in Equation (19.5), we observe that the Kuramoto system can be recovered as a special case of our agent network, by assuming state-independent agent dynamics, i.e., $h(x, u) = u$, setting the pairwise interaction potential to $g(x, y) = \sin(y - x)$, coupling the agents globally, i.e., $\mathcal{B}(x) = \mathcal{X}$, and applying a linear control policy, $\pi_\theta(Y) = \theta \cdot Y$. In this context, the control parameter θ takes over the role of the coupling constant K . However, from a control-theoretic perspective, the agent model is more general in the sense that it is not restricted to linear control policies our sinusoidal observations but it can execute nonlinear policies that operate on arbitrary observational features. This gives rise to a broader class of system behaviors, as will be demonstrated in the subsequent sections.

Simulation Setup

For all presented experiments, we use the basic simulation setup listed below, if not explicitly stated otherwise. The simulation parameters are hand-selected and adapted to the considered scenario, in order to ensure numerical stability of the continuum approximation [Sch03] and to provide a sufficient number of learning iterations for the policy gradient algorithm to converge. Note that all experiments except one are conducted using continuum-based simulations. Nevertheless, we will conveniently specify all model components in terms of agent-based quantities; the corresponding continuum model follows accordingly, as described in Chapter 20. In this spirit, we will generally speak of “the agents” when we discuss our results.

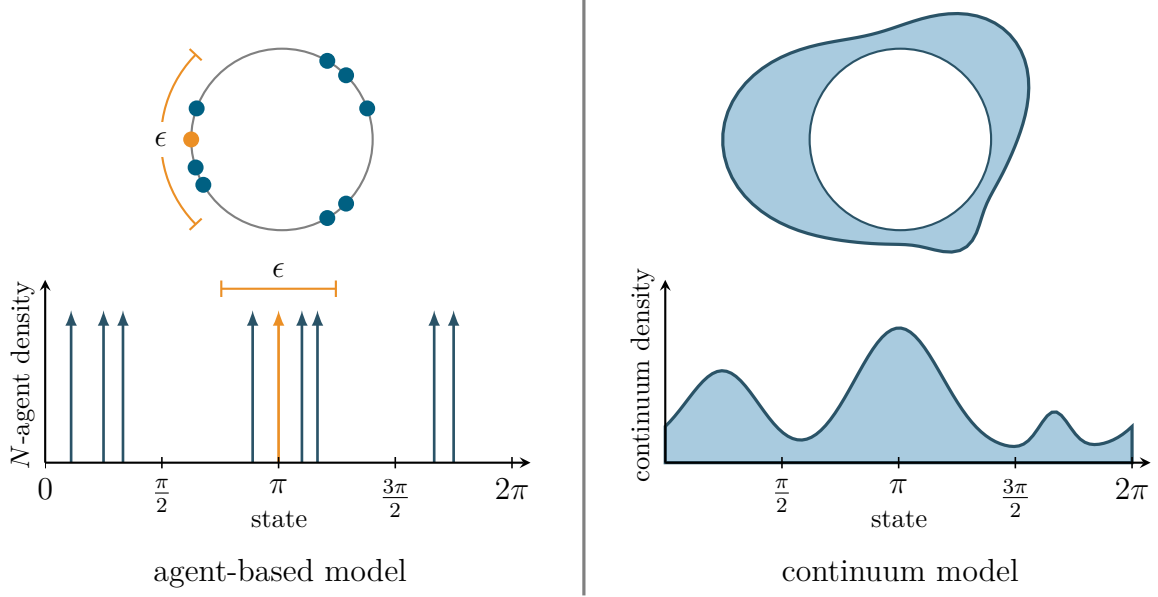


Figure 22.1: Illustration of the agent-based Kuramoto model (left) and the continuum version (right). **Top:** Distribution of the agents/the continuum density on the ring. **Bottom:** Corresponding N -agent density/continuum density on the state space.

System Dynamics The continuum dynamics in Equation (20.3) are approximated numerically using an explicit finite difference method that accounts for the periodic boundary conditions on the ring. Herein, we use a temporal resolution of $\Delta t = 0.01$ and a spatial resolution of $\Delta x = \frac{2\pi}{100}$, resulting in a discretization of the state space $\mathcal{X} = [0, 2\pi)$ into 100 bins. The initial continuum density $\rho(\cdot, 0)$ is distributed homogeneously with small perturbations that break the state symmetry. Further, we use a diffusion coefficient of $D = 0.1$ and a maximum simulation time of $T = 5$.

Learning For the exploration policy of the policy gradient algorithm (Algorithm 3), we use a standard deviation of $\sigma = 0.2$. The gradient is estimated based on $Q = 10$ parameter samples, where we threshold all parameters to the range $[-1, 1]$ after the update to bound the agents' local control signals. For each sample, we perform a single system rollout ($S = 1$) in order to realize a fair comparison of both model types.* During the learning period, the parameters are updated 50 times using a learning rate of $\alpha = 0.2$. Note, however, that the final system performance could be achieved much earlier in most cases.

* Recall that the continuum model requires only one system rollout (see Section 20.3).

22.1 Aggregation

As an introductory example, we consider a simple aggregation task in which we teach the agents to accumulate as fast and close as possible. For this purpose, we adopt the original Kuramoto dynamics, i.e., $g(x, y) = \sin(y - x)$ and $\mathcal{B}(x) = \mathcal{X}$, which we extend with a nonlinear system controller π_θ , i.e.

$$dX_n(t) = \pi_\theta \left(\frac{1}{N} \sum_{m=1}^N \sin(X_m(t) - X_n(t)) \right) dt + dW_n(t).$$

The controller π_θ is parametrized in the form of a radial basis function (RBF) network whose component amplitudes are defined by the entries of θ , i.e.

$$\pi_\theta(Y) = \sum_{c=1}^C \theta_c \cdot \exp \left\{ - \left(\frac{Y - q_c}{\gamma} \right)^2 \right\}.$$

Herein, $\{q_c\}_{c=1}^C$ are the mode locations of the RBF network, which we place on a regular grid on the agents' observation space $\mathcal{Y} = [-1, 1]$ (see left part of Figure 22.2). Note that the finiteness of \mathcal{Y} arises from the sinusoidal shape of the applied interaction potential g . For the experiment, we use a total of $C = 10$ basis functions whose width is chosen according to the grid spacing, i.e., $\gamma = \frac{2\pi}{C-1}$.

During learning, desired system behavior is rewarded by a global reward signal $r^G(t)$, which, in this experiment, takes the form of the system's order parameter [Kur75], i.e.

$$r^G(t) \triangleq \frac{1}{N} \left| \sum_{m=1}^N \exp(iX_m(t)) \right|.$$

Note that i refers to the imaginary unit in the above equation. Figure 22.2 shows the outcome of 100 Monte Carlo runs of the experiment. The left subfigure depicts the learned control parameters and the resulting system policy; the right subfigure illustrates the induced behavior. The results show that the algorithm is able to reliably learn a system policy that solves the aggregation task.

22.2 Partial Observability

Considering the system process from a global perspective, the optimization task can be effectively treated as a single-agent control problem (see Section 19.4.1). Nevertheless, the internal distributed character of the system remains and the degree of local observ-

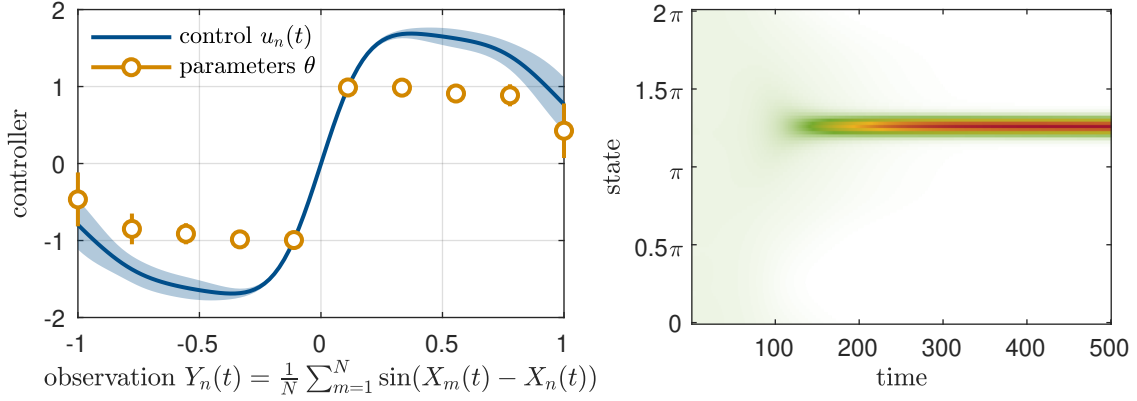


Figure 22.2: Results for the aggregation task in Section 22.1, based on 100 Monte Carlo runs. **Left:** Learned system controller. The orange circles and bars indicate the empirical mean values and standard deviations of the learned control parameters used for the RBF policy network. The blue curve represents the mean control policy defined by those parameters, again shown with the corresponding standard deviation. Positive control values let the executing agent drift toward larger angles, negative values cause a drift in the opposite direction. **Right:** Example system trajectory generated from a random, approximately uniform initial agent density using the mean control parameters from the left subfigure. The trajectory corresponds to a temporal rollout of the continuum density as illustrated in Figure 22.1. Red color indicates high density values.

ability crucially affects the flow of information in the system. The goal of this section is to investigate how different observation modalities of the agents influence the learning process of the network. To this end, we re-run the experiment from Section 22.1 in a partially observable environment, by restricting the interaction of the agents to a limited range $\epsilon \in (0, \pi]$, i.e.

$$\mathcal{B}(x) \triangleq \{y \in \mathcal{X} : |\angle(x, y)| \leq \epsilon\}.$$

Herein, $|\angle(x, y)|$ denotes the absolute angular distance between state x and state y . Note that the maximum value of $\epsilon = \pi$ recovers the setting in Section 22.1.

As before, we use the global order parameter $r^G(t)$ to train the system but consider an extended simulation period of $T = 20$ to account for the increased difficulty of the task. The learning result is depicted by the solid orange line in Figure 22.3, which displays the final order parameter $r^G(T)$ of the trained system for different interaction ranges. The plot shows that the system performance quickly breaks down in small-range interaction regimes. This is also reflected in the right subfigure, which reveals the system’s inability to learn a functioning control policy for small values of ϵ . The obtained result is not surprising as, for small ϵ , the information content available to the agents is not

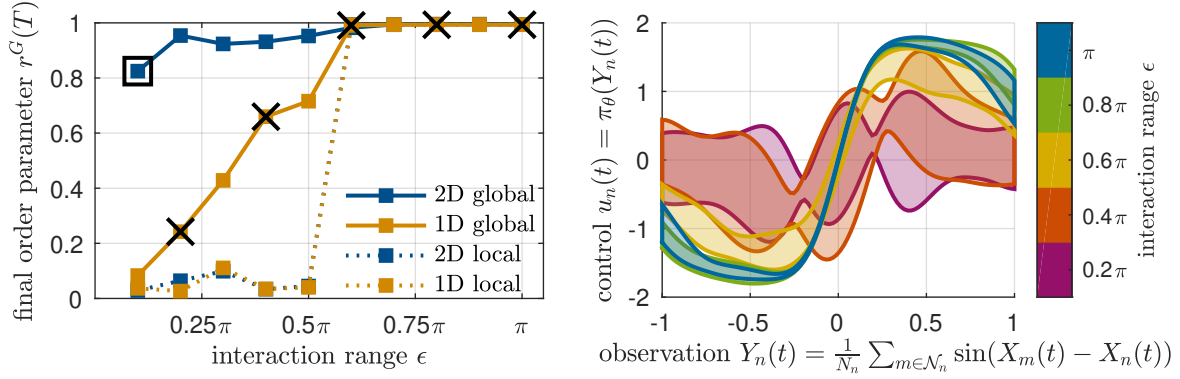


Figure 22.3: Results for the partially observable setting in Section 22.2, based on 20 Monte Carlo runs. **Left:** Aggregation performance for different interaction ranges ϵ , measured in terms of the network’s final order parameter. Legend: global/local indicates the type of reward signal used during training. 1D refers to the sinusoidal observation model, 2D refers to the augmented model that allows the agents to additionally sense the relative agent mass in their vicinity. For large ϵ , the state synchronization is always successful, whereas for small ϵ , a functioning aggregation strategy is only found if the network is trained with global reward information. **Right:** Standard deviation intervals of the learned control policies (centered around their mean values) for the 1D global setting. The colored areas are analogous to the blue area shown in the left part of Figure 22.2, but correspond to different interaction ranges ϵ , as marked by the black crosses in the left subfigure. For small ϵ , the system is unable to learn a functioning aggregation policy, as reflected by the corresponding order parameters shown on the left. For $\epsilon = \pi$, the result from Figure 22.2 is recovered.

sufficient to solve the aggregation task reliably: due to the limited observation range, it is impossible for the agents to decide locally in which direction to move in order to form a single coherent aggregation instead of creating multiple smaller ones.

An interesting question is, therefore, if the agents can develop a functioning strategy when they are provided with additional information. To verify this hypothesis, we equip the agents with extended observation capabilities that allow them to determine the relative agent number in their vicinity, giving rise to the following two-dimensional local observation,

$$Y_n(t) \triangleq \begin{bmatrix} \frac{1}{N_n} \sum_{m \in \mathcal{N}_n} \sin(X_m(t) - X_n(t)) \\ \frac{N_n}{N} \end{bmatrix}. \quad (22.1)$$

While the resulting observation model is not directly consistent with Equation (19.2), the above control input $Y_n(t)$ can be computed locally by agent n from its received observational features $\xi(X_n(t), X(t))$ if we assume that the agent is aware of the total network size N (Remark 6). The latter assumption is fulfilled if we treat N as an

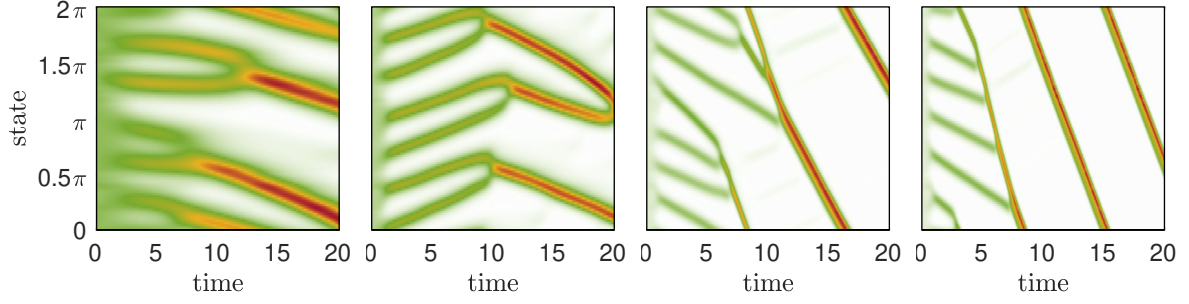


Figure 22.4: Learning progress in the partially observable environment (Section 22.2) when the system is trained with global reward feedback, using the two-dimensional observation model in Equation (22.1) and an interaction range of $\epsilon = \frac{\pi}{10}$. The setting is indicated by the black square in Figure 22.3. Red color indicates high continuum densities. The controller learns an aggregation strategy that exploits the local agent mass to accumulate all agents by assigning different drift velocities.

observable environmental feature (Remark 4), i.e.

$$g(x, y) \triangleq [\sin(y - x), N]^\top. \quad (22.2)$$

Since the resulting observation space is now two-dimensional, i.e., $\mathcal{Y} = [-1, 1] \times [0, 1]$, we use a bivariate RBF network to parametrize the system policy, i.e.

$$\pi_\theta(Y) \triangleq \sum_{c=1}^C \theta_c \cdot \prod_{k=1}^2 \exp \left\{ - \left(\frac{Y_k - q_{c,k}}{\gamma_k} \right)^2 \right\}.$$

Note that the subscript k here indicates the observation dimension and not the agent index. As before, we place the modes $\{q_c\}$ of the basis functions on a regular grid on the observation space by discretizing the first dimension into 10 center positions and the second dimension into 4 positions, resulting in a total of $C = 40$ control parameters to be learned.

By exploiting the additional state information, the agents are now able to solve the aggregation task for arbitrary interaction ranges, as is indicated by the solid blue line in Figure 22.3. It is particularly interesting to inspect closer the learning progress and the final aggregation strategy found by the algorithm, illustrated in Figure 22.4. At first, the policy performs no better than the one trained on the one-dimensional observation model—the controller only manages to aggregate the agents locally. However, after a few iterations, the controller learns to exploit the additional state information contained in the neighborhood size: it assigns different drift velocities (and directions) to agent constellations of different sizes to form a rotating group of agents that eventually absorbs

all smaller agent aggregations. This strategy is then optimized toward the end of the learning period. It is worth mentioning that a similar behavior was discovered by Hamann [Ham14] in the context of a very different learning problem, where the goal of the agents was to predict their local observations one step ahead.

As a final variant of the experiment, we replace the global reward signal $r^G(t)$ with the fused signal $r^L(t)$ (Equation 19.9), assuming that the central critic system has no access to the global system state. For this purpose, we let each agent n compute its own “local order parameter”, i.e.

$$r_n(t) \triangleq \frac{1}{N_n} \left| \sum_{m \in \mathcal{N}_n} \exp \left\{ i \left(X_m(t) - X_n(t) \right) \right\} \right|,$$

which measures the local alignment of the agent relative to its neighbors. Accordingly, we extend the interaction potential g by a dimension of the form $\exp\{i(y - x)\}$, to provide the agents with the necessary state information. Note that, for $\epsilon = \pi$, the locally computed reward $r^L(t)$ reported to the critic coincides with the global signal $r^G(t)$ that we used in the centralized setting with known global system state (Section 22.1).

With the entire reward information being computed locally at the agent level, we compare the resulting system performance to the centralized setting, again by measuring the final order parameter of the system (dashed lines in Figure 22.3). The results reveal that, for small ϵ , the aggregation now fails again, even for the augmented observation model in Equation (22.1) that additionally captures the relative neighborhood size. This time, however, the reason for the failure is different and can be traced back to the learning phase—the local state information provided to the agents is, in fact, sufficient to solve the problem, as we have seen just before. The problem is rather that the critic cannot tell a locally aggregated system state from a globally aggregated one based on the reported feedback signal because both system states result in similar reward signatures when measured locally. Consequently, the learning algorithm is unable to develop a functioning strategy that favors one of the two system states.

The result gives rise to the following interesting conclusion: while the local state information of the agents can be sufficient for *executing* a certain task, it might not be sufficient for *learning* the task in the first place. Yet, the example also demonstrates that a global system goal may be still achievable through local interaction if the behavior is learned under global supervision. This underpins the idea of guided learning presented in [HŠN17] and motivates the use of a centralized feedback architecture to acquire new

skills for a task whose execution can be finally conducted in a decentralized manner. Note, however, that such a learning paradigm not only requires the existence of a central critic system but also relies on a bidirectional communication channel between the critic and the agents, which is often only available in a simulated environment. Hence, if the learning phase is to be conducted on the real system, a decentralized architecture might be the only viable option (see *Discussion and Outlook*).

22.3 Aggregation and Localization

As a last experiment, we consider a combination of the aggregation and the localization problem, where we teach the agents to collectively approach a target position. As part of the experiment, we investigate how a finite-size agent system performs compared to the continuum model.

The basic simulation setup is the one described in Section 22.1 but we replace the initial state distribution with a Gaussian mixture consisting of two balanced mixture components at 0.3π and 0.7π . In addition to the aggregation reward, we pay a target reward at the end of each training episode that is proportional to the agent mass contained in the state discretization bin located at $\frac{3\pi}{2}$, i.e., in the interval $[\frac{3\pi}{2} - \Delta x, \frac{3\pi}{2})$. The relative weight of that reward compared to the aggregation reward is chosen as 50:1 ($\hat{=} \frac{\pi}{\Delta x} : 1$), meaning that the reward paid for accumulating all agent mass inside the target region at the end of an episode is fifty times higher than the reward paid for perfect state synchronization throughout the whole episode. While this ratio is hand-selected, we observed that the learning results are largely insensitive to the specific choice of weights in a wide range. In order to be able to locate the newly introduced target reward, we let the agent additionally sense their absolute position in space and set

$$g(x, y) \triangleq [\sin(y - x), x]^\top. \quad (22.3)$$

Accordingly, we use a two-dimensional RBF representation for the policy based on 6×6 grid positions (indicated by the black crosses in Figure 22.5), resulting in a total of $C = 36$ control parameters.

The rationale behind the described setup is the following: while each agent can contribute to a high reward by approaching the target region along the shortest path independently of the other agents, a higher total return can be achieved if all agents move coherently and additionally exploit the aggregation reward. However, the task is constructed in such a way that the shortest paths differ for both initial agent groups, and hence, the agents

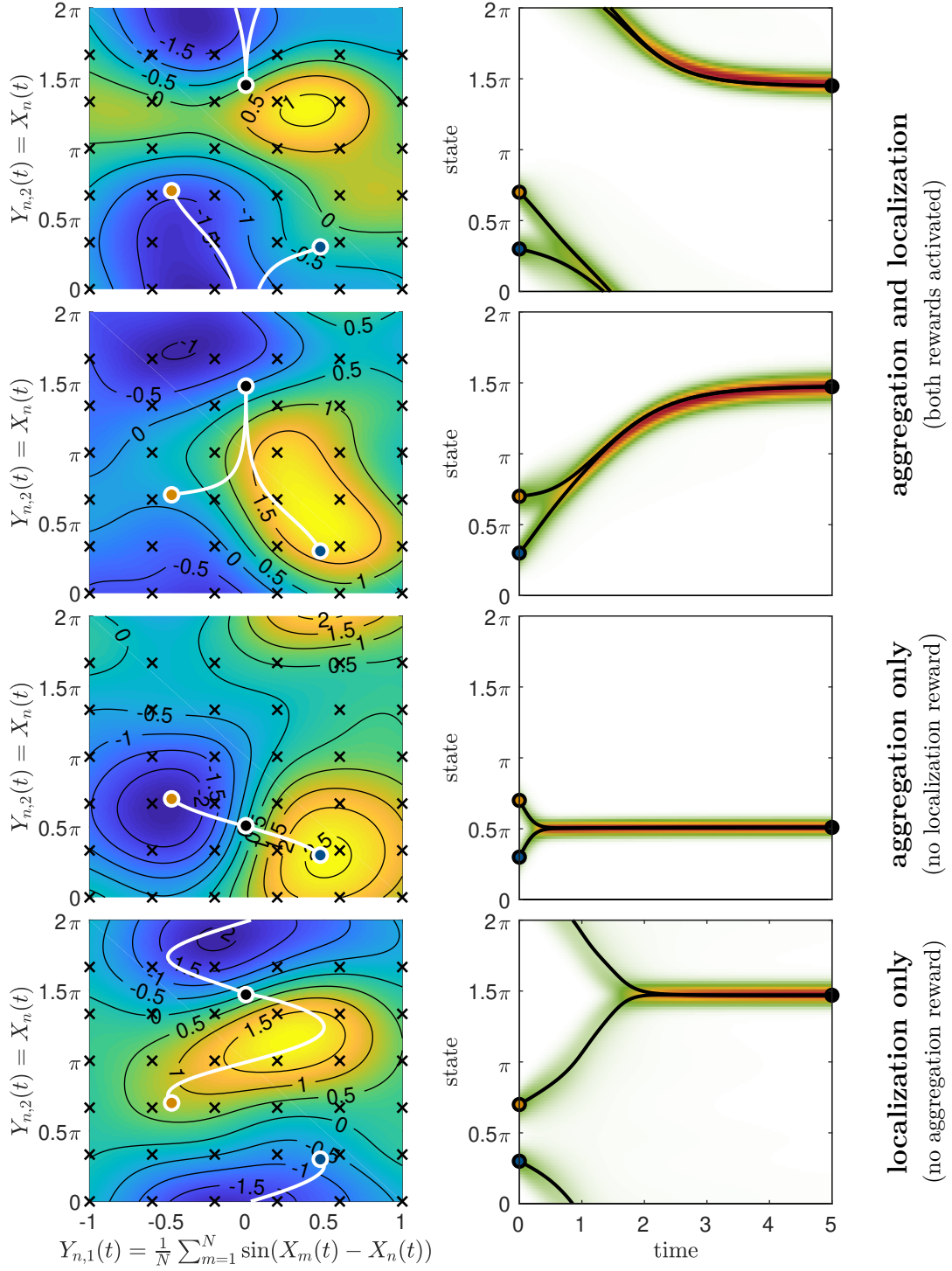


Figure 22.5: Simulation results for the aggregation-localization task (Section 22.3). **Left column:** Example policies found for the different reward settings described on the right. The black crosses mark the center positions of the RBF network. Note that the second dimension (vertical axis), which indicates the position of the agent, is treated cyclically in the parametrization. **Right column:** Resulting system trajectories. Red color indicates high density values. In addition to the continuum density, we further show the trajectories of a noise-free two-agent system executing the learned continuum policy (black lines). The agent states are initialized at the density modes. The same trajectories are projected onto the agents' observation space in the left column (white lines).

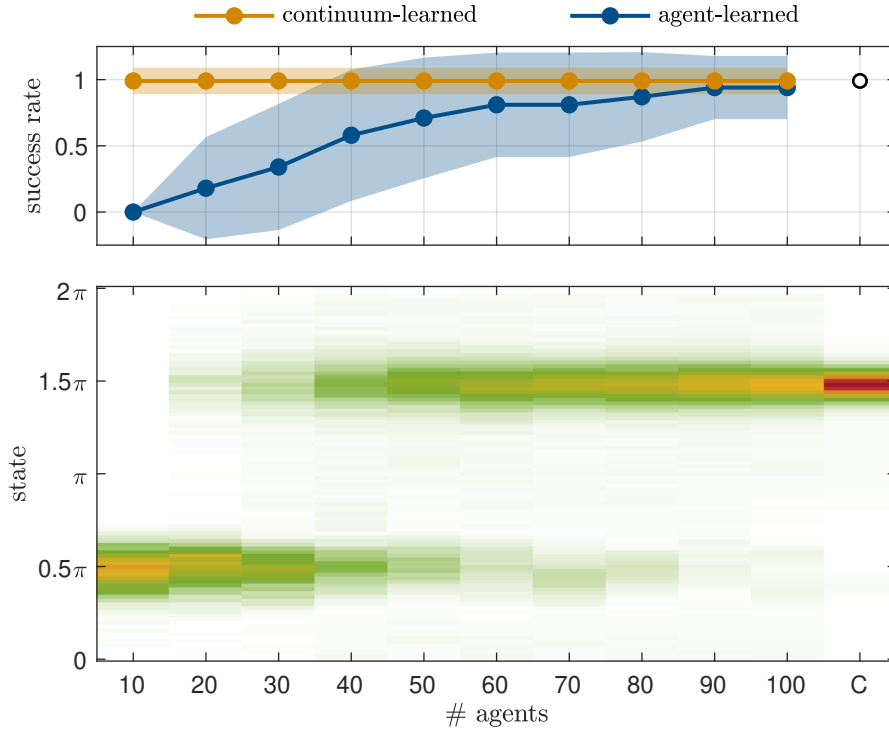


Figure 22.6: Performance comparison of different control policies learned by the continuum system and by agent networks of various sizes, based on 100 Monte Carlo runs of the aggregation-localization experiment (Section 22.3). Both figures share the same abscissa, which indicates the size of the *test system*. Herein, “C” stands for the continuum model. **Top:** Success rate (and corresponding standard deviation) of learning a policy that drives the agents toward the target region located at $\frac{3\pi}{2}$. The color of the line indicates whether the policy was *learned* using the continuum system or using an agent-based network. In the latter case, the training system was of the same size as the test system, as indicated by the abscissa value. For the black circle, both testing and training took place in the continuum domain. **Bottom:** Corresponding final state distributions. The shown values correspond to the blue line in the top figure (agent-learned policies) and to the black circle (continuum-learned policy).

need to break the initial state symmetry in order to decide for a common movement direction. As it turns out, both optimal strategies (clockwise and counterclockwise movement) can be observed in the experiments (Figure 22.5). Moreover, by deactivating either of the two rewards, we can teach the system to focus on only one of the two tasks.

22.3.1 Continuum Modeling versus Agent-Based Modeling

As explained in Section 20.3, a continuum formulation can be advantageous in situations where the variance of the measured return is high (e.g., when the environment contains a localized reward as in the last example), which requires a high number of agent-based system rollouts to reliably estimate the policy gradient. To demonstrate this effect, we

compare the learned behavior of the continuum model with that of the corresponding agent-based system for different network sizes.

Figure 22.6 shows the final agent distributions of all tested systems, averaged over 100 Monte Carlo runs. As expected, for small network sizes, the agent-based systems mainly focus on the easier recognizable aggregation reward and the agents are distracted from the target region, losing much of the achievable return. In contrast, the continuum system learns the optimal strategy with an almost perfect success rate, due to its more reliable gradient estimation.

It is worth mentioning that the learned continuum policy can be readily applied to the agent-based systems, since both model formulations are compatible with each other. When equipped with the continuum policy, all agent networks indeed achieve the same performance level as the continuum system (orange line in Figure 22.6). Example trajectories for a two-agent system are shown by the black and white lines in Figure 22.5. For a real-world problem, this motivates the idea of training the system off-line using a continuum simulation and deploying the learned policy to the real system of agents that finally executes the collective task.

23

Summary

In this part, we presented a continuum framework for modeling the collective behavior of large groups of interacting agents in a reinforcement learning context. The proposed framework offers a computational strategy for handling large-scale distributed control problems and provides a promising alternative to the classical agent-based paradigm. Simulation results on a Kuramoto-type of system demonstrated the capabilities of the framework in a number of collective control tasks. Most notably, our results show that the approach can lead to better performing system policies than obtained through an equivalent agent-trained model. Apart from multi-agent coordination, the presented methodology is also useful for learning single-agent control (Section 20.4); hence, we expect that our findings can be reproduced in a variety of other reinforcement learning scenarios.

Discussion and Outlook

In this dissertation, new modeling techniques were developed that facilitate the detection, analysis and prediction of behavioral patterns in observed single-agent and multi-agent decision-making processes. For single-agent scenarios, two coexisting modeling paradigms were discussed that formulate the inference problem at different conceptual levels, i.e., via intentional and subintentional reasoning. For each paradigm, a separate inference framework was proposed that addresses the problem by constructing a subtask representation of the shown behavior at the respective level, i.e., using local subgoal or policy encodings. In the multi-agent domain, the problem of large-scale decision-making in homogeneous agent networks was investigated. To this end, a new system model was introduced that provides an efficient solution to the IRL problem, and a continuum framework was derived that facilitates the network simulation in large-scale regimes.

Aside from the above-mentioned advances, the thesis raises a number of questions that can be explored in future studies. Some important issues are summarized below.

Single-Agent Modeling

In Part I, we focused on subintentional modeling and developed an inference framework that allows to capture the behavior of an agent directly at the decision level. While the proposed model can handle arbitrary stochastic policies, the current formulation is restricted to problems with finite action spaces. The reason for this is that the proposed policy representation was used to directly assign the probability masses of the model's action distribution in different parts of the state space (Equation 4.2), which renders a transition to continuous action spaces difficult. A natural extension is hence to introduce a shared parametric form for the local policies that allows to synthesize continuous action profiles through an appropriate local coupling of the underlying parameters.

In Part II, we presented two architectures for modeling the intentions of an agent through subgoal-based encodings of the observed behavioral strategy. The current limitation of this approach is that both architectures are restricted to finite state and action spaces, since the construction of the likelihood model requires knowledge of the optimal state-action value functions for all potential subgoal locations (Equation 10.3). While the subgoal principle carries over straightforwardly to continuous metric spaces, the computation of the state-action values becomes difficult in such domains. Fortunately, for BNIRL, there exist several ways to approximate the likelihood [MCH13] and the

same concepts apply equally to the proposed ddBNIRL framework. Thus, an interesting future study would be to compare both approaches on larger problems where a simple discretization of the environment becomes infeasible.

For both approaches, a key challenge in the design of the models was the need to generalize the observed behavior from limited amounts of demonstration data. To address the problem, the proposed framework exploits the local dependencies of the demonstrations through the use of an appropriate distance metric, resulting in a context-dependent processing and prediction of the behavior. For the experimental study conducted in this work, we took advantage of the properties of the ddCRP to construct a non-exchangeable distribution over partitioning structures that allowed to infer the underlying local behavioral patterns. From a practical point of view, this approach enabled a simple consideration of the context information in the form of pairwise distances, where the required distance metric could be furthermore derived naturally from the system environment (Section 10.2.1). On the negative side, the application of the ddCRP model in continuous domains required us to work with a reduced state space representation, in order to handle the lacking marginal invariance of the model (Section 4.3). While both approaches showed decent performance in the considered experiments, it remains unclear if—and to what extent—this solution affects the predictive power of the models. A first step to answering these questions would be to consider a variant of the model that takes into account the context of the demonstrations (i.e., by retaining a non-exchangeable partitioning distribution) but additionally obeys marginal invariance [FW15]. A viable option to construct such a model would be by thinning a suitable random measure [Fot+13].

Swarm Modeling

In Part III, we developed an IRL framework to infer the local reward function of a homogeneous agent network based on observations of the network’s dynamics. In the design of the framework, we have tacitly assumed that the global network behavior can be realized in terms of locally informed interactions between the agents. While this assumption is trivially fulfilled for self-organizing systems (which naturally operate in a decentralized manner), we cannot generally exclude the possibility that—in an arbitrary expert system—the agents are instructed by a central mediator that has access to the global network state. This brings us back to the following fundamental problems: under what conditions is it possible to reconstruct a given system behavior based on local information? What are the corresponding local policies? In the literature, these questions are

often summarized under the term *micro-macro-link* [Bra+13], which refers to the questions of how a particular behavior at the agent level (the microscopic scale) translates into the emergent properties of the collective (the macroscopic scale) and, vice versa, how a certain global phenomenon can be encoded in terms of local rules. The IRL method presented in this work can be regarded as a computational approach to the latter problem but it does not answer the question in which cases the network behavior is reconstructible in the first place. In particular, it remains unclear how well a centralized solution can be approximated by optimizing local objectives if an exact reconstruction is impossible. A partial answer is given in [Šoš+17b], where we discuss the relationship between the local and global value functions of a system and establish a link between the corresponding characteristic reward function classes; yet, the overall problem remains unsolved.

A promising concept to bridge the gap between the two extremes of working with global state information or coping with incomplete local observations is *guided learning* [HŠN17]. Herein, the idea is to train a decentralized system under global supervision, to ease the learning process and to develop cooperative strategies that successfully handle the partial observability at the agent level. An example of this principle could be observed in Section 22.2, where a centralized reward mechanism enabled the agents to develop a local strategy that solves the aggregation problem based on partial state information. For the development of more complex strategies, the systematic use of the agents' local observation histories should be investigated (see also discussion below).

Continuum Modeling

In Part IV, we presented a continuum framework that facilitates the simulation of homogeneous agent networks in large-scale regimes. While the obtained results are promising, the presented experiments should be understood as a proof of concept insofar as we used a simple test system with one-dimensional state space for demonstration purposes. The proposed methodology applies, of course, to higher-dimensional problems; however, a direct application to systems with more than a few state dimensions quickly becomes intractable due to the simplistic state discretization approach followed in this work. For more complex scenarios, it is thus necessary to resort to alternative approximation schemes with better scaling properties. One option is to consider hybrid approaches that combine the strengths of continuum modeling and agent-based simulation. In such an approach, most state dimensions would be treated using an agent-based dynamics model while only a few dimensions, which require a fine-grained resolution of states, would be described as a continuum (compare Rao-Blackwellized particle filters [DGA00]). An alternative option is to use a more sophisticated and scalable approximation of the continuum

density itself, in order to avoid the costly discretization of the system state space in the first place. One possibility is to make use of spectral approximation techniques, i.e., by expanding the continuum density using a small set of basis functions with time-dependent coefficients, which can be computed by an ordinary differential equation solver [FF15].

Apart from computational aspects, two learning-related shortcomings of the current model formulation should be discussed. The first is the use of reactive control policies, which compute the local control signals of the agents based on the current observation input (Remark 5). In order to make informed decisions in complex environments, it can be necessary for the agents to consider (at least parts of) their observation histories, to account for the non-Markovianity of the local observation processes [KLC98]. In theory, such an extension can be formulated for the continuum model by expanding the observation field $\bar{y}(x, t)$ to a more general field of observation histories; however, such an extension goes beyond the scope of this thesis and we leave it for future work. The second shortcoming is the requirement of a centralized learning architecture, which preserved the homogeneity of our system during the learning phase by triggering global policy updates for all agents. Unfortunately, this update strategy does not allow to simulate the learning processes of natural decentralized networks, in which agents learn from local state information and share their experiences locally. A promising future direction is to combine the continuum principle with diffusion-based learning strategies, which spread the gathered local experiences of the agents throughout the network [Mac+15]. Unfortunately, any local policy update in such a learning scheme would inevitably destroy the homogeneity of the system. A possible solution to this problem is to treat the control parameter explicitly as part of an agent's local state, which is tantamount to equipping the agents with an *internal state variable*. Also, since the continuum model only requires homogeneity *within* an agent population, further studies could be carried out by modeling systems that consist of different *population types*, such as predator-prey systems [EE93].

Lastly, we see a promising application of the framework in combination with multi-agent learning paradigms that make explicit use of global state information, such as the guided learning approach described in [HŠN17]. Representing the global network state in some ordered object like the state matrix $X(t)$ (Chapter 19) always comes with the drawback that the inherent symmetries of the system, like its permutation invariance, are hidden by the representation. This renders any centralized optimization process highly inefficient because the learning algorithm will not be able to exploit any of the latent system symmetries. The continuum representation, which provides a permutation-invariant description of the system state, offers an elegant and natural solution to this problem.

APPENDICES

Marginal Invariance and Policy Prediction

When we extended our policy recognition framework to uncountably infinite state spaces using a reduced model formulation in Section 4.3, we inevitably arrived at the following questions: by describing the expert behavior only along observed trajectory states, what does the resulting model imply for the remaining parts of the state space? Can we still use it for predicting the associated local policies?

When investigating these questions from a probabilistic perspective, one quickly stumbles upon a property known as *marginal invariance* [BF11; FW15] (sometimes also referred to as *marginalization property* or simply *consistency* [RW06]). The property ensures that the corresponding model is consistent in the sense that it always provides the same marginal distributions for any fixed subset of its variables, irrespective of the model size. In other words, a marginally invariant policy model always yields the same behavioral representation for a given set of target states, even if we include additional parts of the system state space \mathcal{S} into our reduced modeling set $\tilde{\mathcal{S}}$ for which there is no demonstration data available.

For the considered spatial models (i.e., the Potts model and the ddCRP-based models), it can be shown that this consistency property is indeed lacking — see [BF11] for a detailed explanation. This means that we cannot expect to get consistent answers when conducting our reduced model inference on two data sets of different sizes, where one data set is a subset of the other. On the contrary, making predictions at new states requires to rerun the Gibbs sampler on the augmented model that includes all target states right from the beginning.

From a practitioner’s point of view, this may cause a dilemma in certain situations. Imagine an on-line policy recognition scenario where we observe an expert controlling the target system. After a certain period of time, we are asked to take over control, using the experience we have acquired during the observation period. Each control command, whether performed by the expert or by us, will trigger a new state transition, meaning that new data points stream in sequentially. In this case, it is impossible to decide in advance which system states to include in the reduced set $\tilde{\mathcal{S}}$ and which not to include. A rigorous approach in the above-described sense would thus require to recalibrate the model after each state transition — a costly operation.

However, it is evident that the resulting data set is naturally divided into two disjoint parts, namely the expert demonstrations and the subsequent states reached during execution of the learned policy. Clearly, transitions occurring after the learning phase should by no means affect our belief about the expert policy, and hence, they should not be considered in our prediction. The easiest way to achieve this is, indeed, to “freeze” the model after the demonstration phase and use the learned behavioral representation to extrapolate the expert knowledge to surrounding states. This can be done, for instance, by retaining the structure of the partitioning prior model to compute the MAP estimates for the corresponding indicator variables at the query states.

In the case of the ddCRP, this coincides with the nearest-neighbor estimate (see Equation 5.3), i.e.

$$\hat{c}_{\text{new}} = \arg \max_{t \in \{1, \dots, T\}} f(\Delta_{t, \text{new}}) = \arg \min_{t \in \{1, \dots, T\}} \Delta_{t, \text{new}}, \quad (\text{A.1})$$

where \hat{c}_{new} is the estimate for the indicator of the new state and $\Delta_{t, \text{new}}$ denotes the distance of that state to the t th trajectory point. Note that the contribution of the self-link parameter ν is ignored in the above equation as it corresponds to the hypothesis that the expert switches to a previously unseen policy, in which case no prediction (other than the one provided by the policy prior model) would be possible.

In summary, one could argue that the simplicity of retaining a finite model structure for modeling infinite state space problems comes at the cost of not being able to construct a consistent posterior predictive distribution. However, the reduced state space approach offers a very uncomplicated and intuitive way to account for the spatial context of the demonstration data, i.e., in the form of pairwise distances. Also, the simulation results in Section 6.1 demonstrate that the reduced model is able to capture the relevant spatial dependencies of a policy sufficiently well to make profound predictions about unseen states. Nonetheless, an interesting future study would be to investigate how the presented distance-based approach compares to non-exchangeable models that obey marginal invariance (see *Discussion and Outlook*).

B

KUKA Robot Experiment

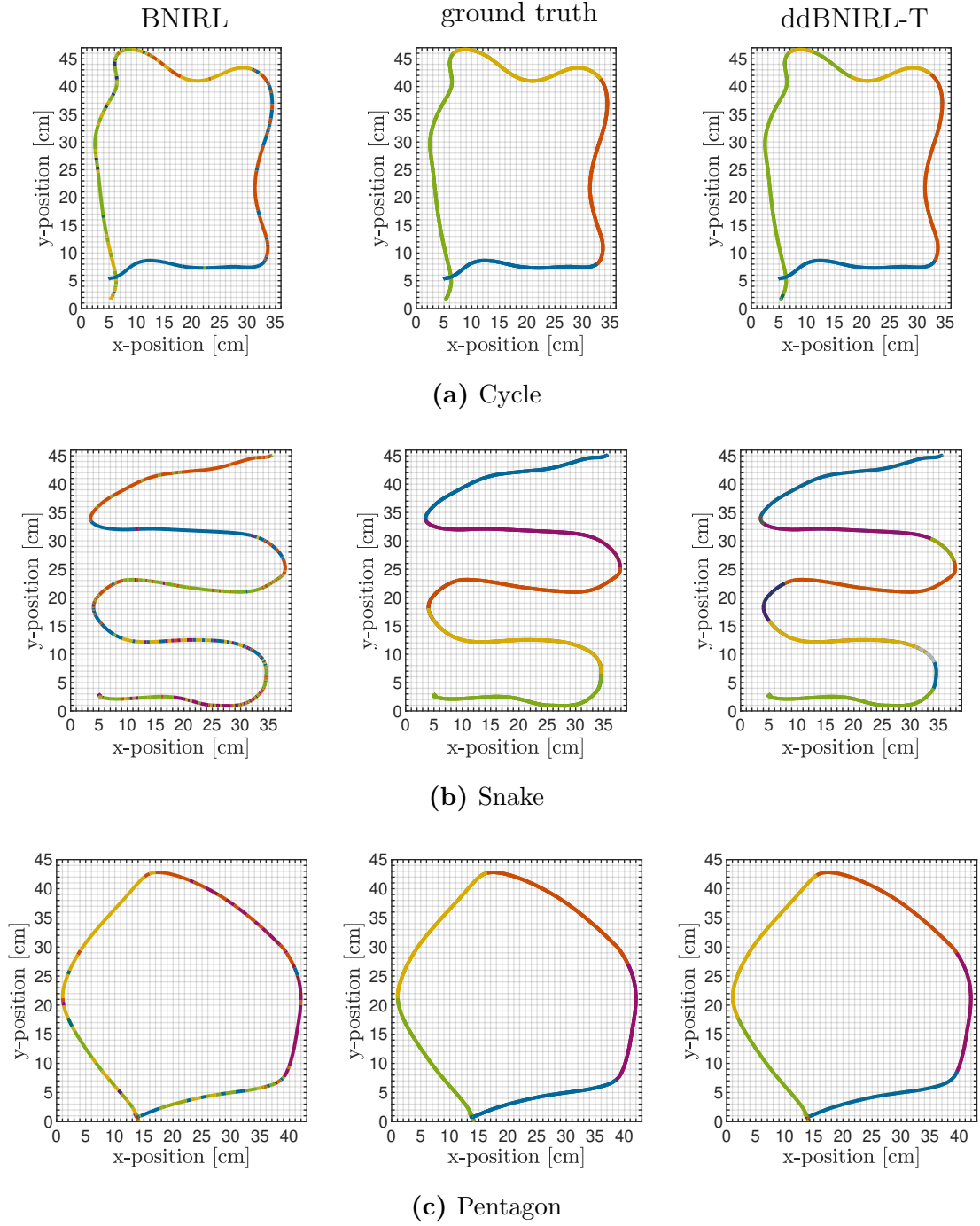
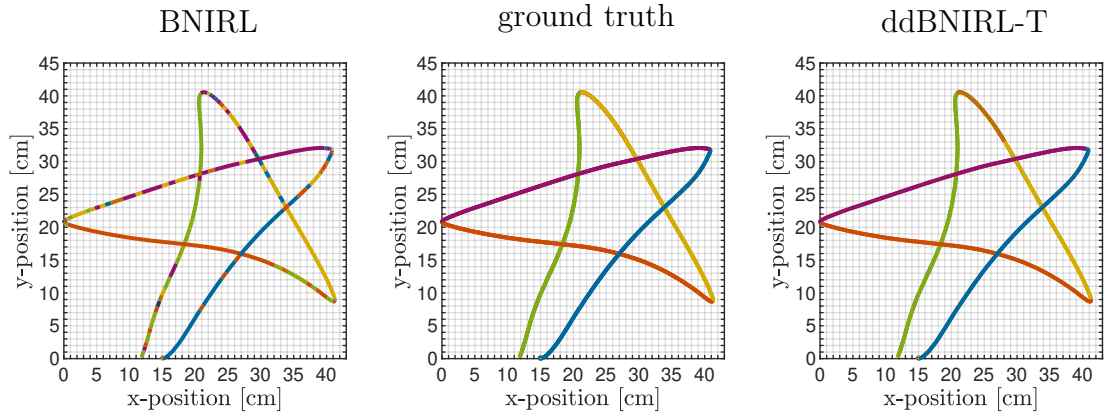
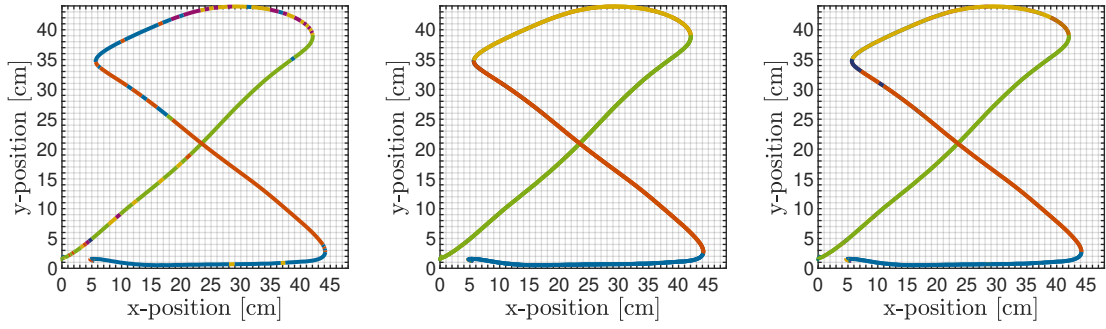


Figure B: Motion sequences without trajectory crossings, which can be represented using a spatial subgoal pattern.

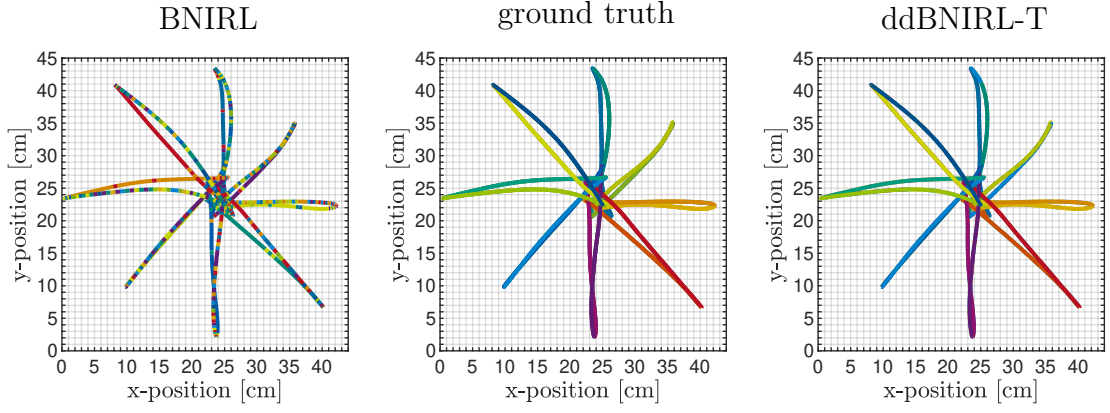


(d) Star

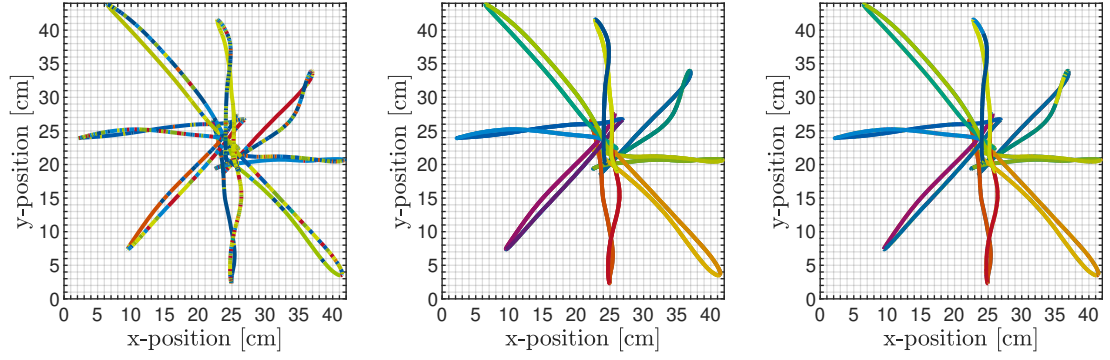


(e) Hourglass

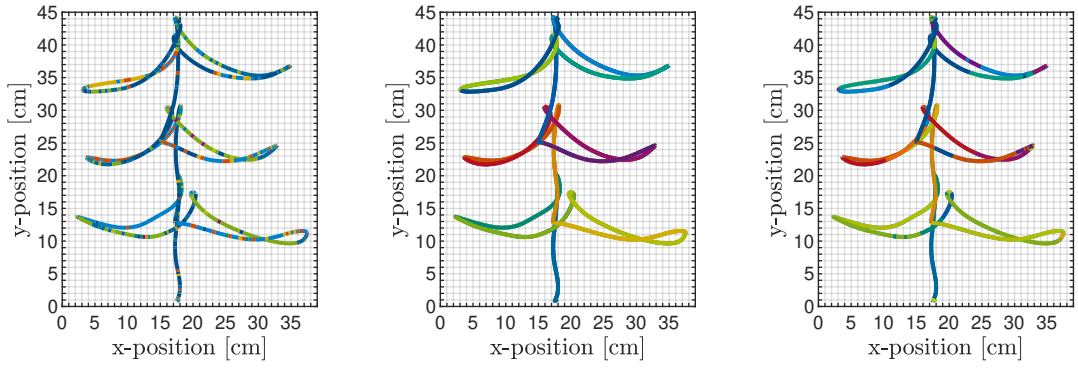
Figure B (continued): Motion sequences with few trajectory crossings, requiring a time-varying subgoal representation.



(f) Flower (Const)



(g) Flower (Var)



(h) Tree

Figure B (continued): Long motion sequences comprising a large number of sub-patterns with overlapping parts that can be only separated by considering the temporal context. Flower (Const): all strokes are performed with the same absolute velocity. Flower (Var): the individual strokes are performed with alternating velocity.

Derivation of the Continuum Equation

In the following, we show how the continuum equation (20.3) can be derived from the agent-based system of stochastic differential equations (19.1) as the number of agents in the network approaches infinity. Herein, we follow the basic steps in [Dea96], which we extend by the necessary control-related quantities.

Our goal is to find an analytic expression for the temporal evolution of the global agent density $\rho^{(N)}(x, t)$ for $N \rightarrow \infty$. We start with an Itô expansion [Kry80] of the stochastic differential equation (19.5), i.e.

$$\begin{aligned} df(X_n(t)) &= \nabla f(X_n(t)) \cdot h(X_n(t), \pi_\theta(\xi(X_n(t), X(t)))) dt \\ &\quad + D\nabla^2 f(X_n(t)) dt + \nabla f(X_n(t)) \cdot dW_n(t), \end{aligned} \tag{C.1}$$

where $f : \mathcal{X} \rightarrow \mathbb{R}$ is a twice-differentiable function. Using the identity

$$f(X_n(t)) = \int_{x \in \mathcal{X}} \rho_n(x, t) f(x) dx, \tag{C.2}$$

which follows from the definition of the single-agent density in Equation (20.2), we rewrite Equation (C.1) as

$$\begin{aligned} df(X_n(t)) &= \int_{x \in \mathcal{X}} \left[\nabla f(x) \cdot h(x, \pi_\theta(\xi(x, X(t)))) dt \right. \\ &\quad \left. + D\nabla^2 f(x) dt + \nabla f(x) \cdot dW_n(t) \right] \rho_n(x, t) dx. \end{aligned} \tag{C.3}$$

Next, we integrate Equation (C.3) by parts, which yields

$$\begin{aligned} df(X_n(t)) &= \int_{x \in \mathcal{X}} \left\{ -\nabla \cdot \left[\rho_n(x, t) h(x, \pi_\theta(\xi(x, X(t)))) \right] dt \right. \\ &\quad \left. + D\nabla^2 \rho_n(x, t) dt - \nabla \cdot \rho_n(x, t) dW_n(t) \right\} f(x) dx. \end{aligned}$$

Note that Identity (C.2) further implies

$$df(X_n(t)) = \int_{x \in \mathcal{X}} d\rho_n(x, t) f(x) dx. \tag{C.4}$$

Comparing Equations (C.3) and (C.4), it follows that

$$\begin{aligned} d\rho_n(x, t) = & -\nabla \cdot \left[\rho_n(x, t) h\left(x, \pi_\theta(\xi(x, X(t)))\right) \right] dt \\ & + D\nabla^2 \rho_n(x, t) dt - \nabla \cdot \rho_n(x, t) dW_n(t). \end{aligned}$$

In order to obtain an expression for the global density, we sum up all agent-based increments, which gives

$$\begin{aligned} d\rho^{(N)}(x, t) &= \frac{1}{N} \sum_{n=1}^N d\rho_n(x, t) \\ &= -\nabla \cdot \left[\rho^{(N)}(x, t) h\left(x, \bar{u}^{(N)}(x, t)\right) \right] dt \\ &\quad + D\nabla^2 \rho^{(N)}(x, t) dt - \nabla \cdot \frac{1}{N} \sum_{n=1}^N \rho_n(x, t) dW_n(t), \end{aligned} \tag{C.5}$$

where we introduced the finite-size control field $\bar{u}^{(N)}(x, t)$, i.e.

$$\bar{u}^{(N)}(x, t) \triangleq \pi_\theta(\bar{y}^{(N)}(x, t)), \tag{C.6}$$

and the underlying observation field $\bar{y}^{(N)}(x, t)$, i.e.

$$\bar{y}^{(N)}(x, t) \triangleq \xi(x, X(t)) = \frac{\int_{\mathcal{X}} \rho^{(N)}(y, t) g(x, y) k(x, y) dy}{\int_{\mathcal{X}} \rho^{(N)}(y', t) k(x, y') dy'}, \tag{C.7}$$

as replacements for the agent-based control and observation signals $\{u_n(t)\}$ and $\{Y_n(t)\}$. Note that Equation (C.7) follows directly from Equation (19.3) using the definition of the N -agent density in Equation (20.1).

As shown by Dean [Dea96], the cumulative influence of the agent-dependent noise terms in Equation (C.5) can be described by a statistically equivalent agent-independent field of noise processes $\bar{W}(x, t)$ with correlation function

$$\mathbb{E}[\bar{W}_i(x, t) \bar{W}_j(y, t')] = 2D\delta_{i,j}\delta_{x,y} \min(t, t'),$$

where $\bar{W}_i(x, t)$ denotes the i th component of the field at position x and time t . Equation (C.5) thus simplifies to

$$\begin{aligned} d\rho^{(N)}(x, t) = & -\nabla \cdot \left[\rho^{(N)}(x, t) h\left(x, \bar{u}^{(N)}(x, t)\right) \right] dt \\ & + D\nabla^2 \rho^{(N)}(x, t) dt + \nabla \cdot \left[\frac{1}{N} \sqrt{\rho^{(N)}(x, t)} d\bar{W}(x, t) \right]. \end{aligned}$$

In the limit as $N \rightarrow \infty$, the stochastic component of this differential equation vanishes and we obtain our final convection-diffusion dynamics for the continuum density, i.e.

$$\frac{\partial \rho(x, t)}{\partial t} = -\nabla \cdot \left[\rho(x, t) h(x, \bar{u}(x, t)) \right] + D \nabla^2 \rho(x, t),$$

where the continuum control field $\bar{u}(x, t)$ and the underlying continuum observation field $\bar{y}(x, t)$ are defined as in Equations (C.6) and (C.7), except that $\rho^{(N)}(x, t)$ is replaced with $\rho(x, t)$.

List of Acronyms

BNIRL	Bayesian nonparametric inverse reinforcement learning
BNIRL-EXT	extended BNIRL
BPR	Bayesian policy recognition
CMP	controlled Markov process
CRP	Chinese restaurant process
ddBNIRL	distance-dependent BNIRL
ddBNIRL-S	spatial ddBNIRL
ddBNIRL-T	temporal ddBNIRL
ddCRP	distance-dependent Chinese restaurant process
dec-POMDP	decentralized partially observable Markov decision process
DPMM	Dirichlet process mixture model
EM	expectation maximization
EMD	earth mover's distance
HMM	hidden Markov model
ICM	iterated conditional modes
IRL	inverse reinforcement learning
LfD	learning from demonstration
MAP	maximum a posteriori
MDP	Markov decision process
MDP\R	Markov decision process without reward function
ML	maximum likelihood
MRF	Markov random field
PEPG	parameter exploring policy gradient
RBF	radial basis function
RL	reinforcement learning
SDM	sequential decision-making
semi-MDP	semi Markov decision process
swarMDP	homogeneous multi-agent Markov decision process

References

- [Abe+00] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. F. Knight Jr., R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss. “Amorphous computing”. In: *Communications of the ACM* 43.5 (2000), pp. 74–82.
- [ACN10] P. Abbeel, A. Coates, and A. Y. Ng. “Autonomous helicopter aerobatics through apprenticeship learning”. In: *International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639.
- [Ald85] D. J. Aldous. “Exchangeability and related topics”. In: *École d’Été de Probabilités de Saint-Flour XIII—1983*. Springer, 1985, pp. 1–198.
- [Amo+14] H. B. Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters. “Interaction primitives for human-robot cooperation tasks”. In: *IEEE International Conference on Robotics and Automation*. 2014, pp. 2831–2837.
- [AN04] P. Abbeel and A. Y. Ng. “Apprenticeship learning via inverse reinforcement learning”. In: *International Conference on Machine Learning*. 2004, p. 1.
- [AN05] P. Abbeel and A. Y. Ng. “Exploration and apprenticeship learning in reinforcement learning”. In: *International Conference on Machine Learning*. 2005, pp. 1–8.
- [Arg+09] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. “A survey of robot learning from demonstration”. In: *Robotics and Autonomous Systems* 57.5 (2009), pp. 469–483.
- [AS97a] C. G. Atkeson and J. C. Santamaria. “A comparison of direct and model-based reinforcement learning”. In: *IEEE International Conference on Robotics and Automation*. 1997, pp. 3557–3564.
- [AS97b] C. G. Atkeson and S. Schaal. “Robot learning from demonstration”. In: *International Conference on Machine Learning*. 1997, pp. 12–20.
- [AT05] G. Adomavicius and A. Tuzhilin. “Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.6 (2005), pp. 734–749.
- [Aum64] R. J. Aumann. “Markets with a continuum of traders”. In: *Econometrica* 32.1/2 (1964), pp. 39–50.

- [Bab+11] M. Babes-Vroman, V. Marivate, K. Subramanian, and M. Littman. “Apprenticeship learning about multiple intentions”. In: *International Conference on Machine Learning*. 2011, pp. 897–904.
- [Bai93] L. C. Baird. *Advantage updating*. Tech. rep. Wright Lab, 1993.
- [Bau+70] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains”. In: *The Annals of Mathematical Statistics* 41.1 (1970), pp. 164–171.
- [BBD08] L. Buşoniu, R. Babuška, and B. De Schutter. “A comprehensive survey of multiagent reinforcement learning”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 38.2 (2008), pp. 156–172.
- [BD94] S. J. Bradtke and M. O. Duff. “Reinforcement learning methods for continuous-time Markov decision problems”. In: *Advances in Neural Information Processing Systems*. 1994, pp. 393–400.
- [Bea05] J. Beal. “Programming an amorphous computational medium”. In: *Unconventional Programming Paradigms*. 2005, pp. 121–136.
- [Bel57] R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [Bes86] J. Besag. “On the statistical analysis of dirty pictures”. In: *Journal of the Royal Statistical Society. Series B* 48.3 (1986), pp. 259–302.
- [BF11] D. M. Blei and P. I. Frazier. “Distance dependent Chinese restaurant processes”. In: *Journal of Machine Learning Research* 12.Aug (2011), pp. 2461–2488.
- [BF97] H. Benbrahim and J. A. Franklin. “Biped dynamic walking using reinforcement learning”. In: *Robotics and Autonomous Systems* 22.3-4 (1997), pp. 283–302.
- [Bha+09] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee. “Natural actor-critic algorithms”. In: *Automatica* 45.11 (2009), pp. 2471–2482.
- [Bil+08] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. “Robot programming by demonstration”. In: *Springer Handbook of Robotics*. Springer, 2008, pp. 1371–1394.
- [Bil99] P. Billingsley. *Convergence of Probability Measures*. Wiley, 1999.
- [BNJ03] D. M. Blei, A. Y. Ng, and M. I. Jordan. “Latent Dirichlet allocation”. In: *Journal of Machine Learning Research* 3.Jan (2003), pp. 993–1022.

- [Bra+13] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. “Swarm robotics: a review from the swarm engineering perspective”. In: *Swarm Intelligence* 7.1 (2013), pp. 1–41.
- [BS02] C. Breazeal and B. Scassellati. “Robots that imitate humans”. In: *Trends in Cognitive Sciences* 6.11 (2002), pp. 481–487.
- [BT03] N. Barberis and R. Thaler. “A survey of behavioral finance”. In: *Handbook of the Economics of Finance* 1 (2003), pp. 1053–1128.
- [BT91] D. P. Bertsekas and J. N. Tsitsiklis. “An analysis of stochastic shortest path problems”. In: *Mathematics of Operations Research* 16.3 (1991), pp. 580–595.
- [Buh+06] J. Buhl, D. J. T. Sumpter, I. D. Couzin, J. J. Hale, E. Despland, E. R. Miller, and S. J. Simpson. “From disorder to order in marching locusts”. In: *Science* 312.5778 (2006), pp. 1402–1406.
- [Bux03] H. Buxton. “Learning and understanding dynamic scene activity: a review”. In: *Image and Vision Computing* 21.1 (2003), pp. 125–136.
- [Cam+06] T. Camilo, C. Carreto, J. S. Silva, and F. Boavida. “An energy-efficient ant-based routing algorithm for wireless sensor networks”. In: *Ant Colony Optimization and Swarm Intelligence*. 2006, pp. 49–59.
- [Ces+17] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu. “Boltzmann exploration done right”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6284–6293.
- [CFL09] C. Castellano, S. Fortunato, and V. Loreto. “Statistical physics of social dynamics”. In: *Reviews of Modern Physics* 81.2 (2009), pp. 591–646.
- [CG93] E. Charniak and R. P. Goldman. “A Bayesian model of plan recognition”. In: *Artificial Intelligence* 64.1 (1993), pp. 53–79.
- [CG95] S. Chib and E. Greenberg. “Understanding the Metropolis-Hastings algorithm”. In: *The American Statistician* 49.4 (1995), pp. 327–335.
- [CK12] J. Choi and K.-e. Kim. “Nonparametric Bayesian inverse reinforcement learning for multiple reward functions”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 305–313.
- [CM06] N. Correll and A. Martinoli. “System identification of self-organizing robotic swarms”. In: *Distributed Autonomous Robotic Systems*. 2006, pp. 31–40.

- [CM95] J. P. Crutchfield and M. Mitchell. “The evolution of emergent computation”. In: *Proceedings of the National Academy of Sciences* 92.23 (1995), pp. 10742–10746.
- [Cou09] I. D. Couzin. “Collective cognition in animal groups”. In: *Trends in Cognitive Sciences* 13.1 (2009), pp. 36–43.
- [Dan+16] C. Daniel, H. Van Hoof, J. Peters, and G. Neumann. “Probabilistic inference for determining options in reinforcement learning”. In: *Machine Learning* 104.2-3 (2016), pp. 337–357.
- [DB02] P. Dayan and B. W. Balleine. “Reward, motivation, and reinforcement learning”. In: *Neuron* 36.2 (2002), pp. 285–298.
- [Dea96] D. S. Dean. “Langevin equation for the density of a system of interacting Langevin processes”. In: *Journal of Physics A: Mathematical and General* 29.24 (1996), p. L613.
- [DFR15] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. “Gaussian processes for data-efficient learning in robotics and control”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2 (2015), pp. 408–423.
- [DGA00] A. Doucet, S. Godsill, and C. Andrieu. “On sequential Monte Carlo sampling methods for Bayesian filtering”. In: *Statistics and Computing* 10.3 (2000), pp. 197–208.
- [Dil+00] R. Dillmann, O. Rogalla, M. Ehrenmann, R. Zöliner, and M. Bordegoni. “Learning robot behaviour and skills based on human demonstration and advice: the machine learning paradigm”. In: *Robotics Research*. Springer, 2000, pp. 229–238.
- [DL09] L. Dufton and K. Larson. “Multiagent policy teaching”. In: *International Conference on Autonomous Agents and Multiagent Systems* (2009).
- [DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society. Series B* 39.1 (1977), pp. 1–38.
- [DNP13] M. P. Deisenroth, G. Neumann, and J. Peters. “A survey on policy search for robotics”. In: *Foundations and Trends in Robotics* 2.1–2 (2013), pp. 1–142.
- [Doy07] K. Doya. “Reinforcement learning: computational theory and biological mechanisms”. In: *HFSP Journal* 1.1 (2007), pp. 30–40.

- [DR11] C. Dimitrakakis and C. A. Rothkopf. “Bayesian multitask inverse reinforcement learning”. In: *European Workshop on Reinforcement Learning*. 2011, pp. 273–284.
- [DT10] K. Dvijotham and E. Todorov. “Inverse optimal control with linearly-solvable MDPs”. In: *International Conference on Machine Learning*. 2010, pp. 335–342.
- [EE93] G. B. Ermentrout and L. Edelstein-Keshet. “Cellular automata approaches to biological modeling”. In: *Journal of Theoretical Biology* 160.1 (1993), pp. 97–133.
- [Eng+13] P. Englert, A. Paraschos, J. Peters, and M. P. Deisenroth. “Model-based imitation learning by probabilistic trajectory matching”. In: *IEEE International Conference on Robotics and Automation*. 2013, pp. 1922–1927.
- [Fer73] T. S. Ferguson. “A Bayesian analysis of some nonparametric problems”. In: *The Annals of Statistics* 1.2 (1973), pp. 209–230.
- [FF15] B. Fornberg and N. Flyer. “Solving PDEs with radial basis functions”. In: *Acta Numerica* 24 (2015), pp. 215–258.
- [FND03] T. Fong, I. Nourbakhsh, and K. Dautenhahn. “A survey of socially interactive robots”. In: *Robotics and Autonomous Systems* 42.3 (2003), pp. 143–166.
- [Fot+13] N. Foti, J. Futoma, D. Rockmore, and S. Williamson. “A unifying representation for a class of dependent random measures”. In: *International Conference on Artificial Intelligence and Statistics*. 2013, pp. 20–28.
- [FR03] D. P. de Farias and B. V. Roy. “The linear programming approach to approximate dynamic programming”. In: *Operations research* 51.6 (2003), pp. 850–865.
- [Fre05] R. A. Freitas. “Current status of nanomedicine and medical nanorobotics”. In: *Journal of Computational and Theoretical Nanoscience* 2.1 (2005), pp. 1–25.
- [FW15] N. J. Foti and S. A. Williamson. “A survey of non-exchangeable priors for Bayesian nonparametric models”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2 (2015), pp. 359–371.
- [GCM05] S. C. Goldstein, J. D. Campbell, and T. C. Mowry. “Programmable matter”. In: *Computer* 38.6 (2005), pp. 99–101.

- [Gha+15] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar. “Bayesian reinforcement learning: a survey”. In: *Foundations and Trends in Machine Learning* 8.5-6 (2015), pp. 359–483.
- [Gra78] M. Granovetter. “Threshold models of collective behavior”. In: *American Journal of Sociology* 83.6 (1978), pp. 1420–1443.
- [Gro+12] I. Grondman, L. Buşoniu, G. A. D. Lopes, and R. Babuška. “A survey of actor-critic reinforcement learning: standard and natural policy gradients”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 42.6 (2012), pp. 1291–1307.
- [Gu+16] Y. Gu, Y. Hashimoto, L. T. Hsu, and S. Kamijo. “Motion planning based on learning models of pedestrian and driver behaviors”. In: *IEEE International Conference on Intelligent Transportation Systems*. 2016, pp. 808–813.
- [Gut+14] T. Guthier, A. Šošić, V. Willert, and J. Eggert. “sNN-LDS: spatio-temporal non-negative sparse coding for human action recognition”. In: *International Conference on Artificial Neural Networks*. 2014, pp. 185–192.
- [Had+16] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan. “Cooperative inverse reinforcement learning”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 3909–3917.
- [Ham14] H. Hamann. “Evolution of collective behaviors by minimizing surprise”. In: *International Conference on the Synthesis and Simulation of Living Systems*. 2014, pp. 344–351.
- [Hay02] A. T. Hayes. “How many robots? Group size and efficiency in collective search tasks”. In: *Distributed Autonomous Robotic Systems*. 2002, pp. 289–298.
- [Hay05] S. Haykin. “Cognitive radio: brain-empowered wireless communications”. In: *IEEE Journal on Selected Areas in Communications* 23.2 (2005), pp. 201–220.
- [HJ09] D. Helbing and A. Johansson. “Pedestrian, crowd and evacuation dynamics”. In: *Encyclopedia of Complexity and Systems Science*. Springer, 2009, pp. 6476–6495.
- [Hjo+10] N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker. *Bayesian Nonparametrics*. Cambridge University Press, 2010.

- [Hoc+16] I. Hochberg, G. Feraru, M. Kozdoba, S. Mannor, M. Tennenholtz, and E. Yom-Tov. “Encouraging physical activity in patients with diabetes through automatic personalized feedback via reinforcement learning improves glycemic control”. In: *Diabetes Care* 39.4 (2016), e59–e60.
- [HŠN16] M. Hüttenrauch, A. Šošić, and G. Neumann. “Guided deep reinforcement learning for swarm systems”. In: *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems*. 2016.
- [HŠN17] M. Hüttenrauch, A. Šošić, and G. Neumann. “Guided deep reinforcement learning for swarm systems”. In: *AAMAS Workshop on Autonomous Robots and Multirobot Systems*. 2017.
- [HŠN18] M. Hüttenrauch, A. Šošić, and G. Neumann. “Local communication protocols for learning complex swarm behaviors with deep reinforcement learning”. In: *International Conference on Swarm Intelligence*. **Accepted**. 2018.
- [HV15] B. Houchmandzadeh and M. Vallade. “Exact results for a noise-induced bistable system”. In: *Physical Review E* 91.2 (2015), p. 022115.
- [HW08] H. Hamann and H. Wörn. “A framework of space-time continuous models for algorithm design in swarm robotics”. In: *Swarm Intelligence* 2.2–4 (2008), pp. 209–239.
- [HZ15] J. Hahn and A. M. Zoubir. “Inverse reinforcement learning using expectation maximization in mixture models”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2015, pp. 3721–3725.
- [Isi25] E. Ising. “Beitrag zur Theorie des Ferromagnetismus”. In: *Zeitschrift für Physik* 31.1 (1925), pp. 253–258.
- [JLK11] A. Jern, C. G. Lucas, and C. Kemp. “Evaluating the inverse decision-making approach to preference learning”. In: *Advances in Neural Information Processing Systems*. 2011, pp. 2276–2284.
- [KBP13] J. Kober, J. A. Bagnell, and J. Peters. “Reinforcement learning in robotics: a survey”. In: *International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.
- [KCC10] P. Kormushev, S. Calinon, and D. G. Caldwell. “Robot motor skill coordination with EM-based reinforcement learning”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 3232–3237.
- [KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

- [KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by simulated annealing”. In: *Science* 220.4598 (1983), pp. 671–680.
- [KII94] Y. Kuniyoshi, M. Inaba, and H. Inoue. “Learning by watching: extracting reusable task knowledge from visual observation of human performance”. In: *IEEE Transactions on Robotics and Automation* 10.6 (1994), pp. 799–822.
- [KKM13] B. M. Kapron, V. King, and B. Mountjoy. “Dynamic graph connectivity in polylogarithmic worst case time”. In: *ACM-SIAM Symposium on Discrete Algorithms*. 2013, pp. 1131–1142.
- [KLC98] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artificial intelligence* 101.1 (1998), pp. 99–134.
- [KLM96] L. P. Kaelbling, M. L. Littman, and A. W. Moore. “Reinforcement learning: a survey”. In: *Journal of Artificial Intelligence Research* 4 (1996), pp. 237–285.
- [Kri+16] S. Krishnan, A. Garg, R. Liaw, L. Miller, F. T. Pokorny, and K. Goldberg. HIRL: hierarchical inverse reinforcement learning for long-horizon tasks with delayed rewards. 2016. arXiv: 1604.06508.
- [Kry80] N. V. Krylov. *Controlled diffusion processes*. Springer, 1980.
- [KS15] V. Kuleshov and O. Schrijvers. “Inverse game theory: learning utilities in succinct games”. In: *Web and Internet Economics*. 2015, pp. 413–427.
- [Kur75] Y. Kuramoto. “Self-entrainment of a population of coupled non-linear oscillators”. In: *International Symposium on Mathematical Problems in Theoretical Physics*. 1975, pp. 420–422.
- [LB95] M. Land and R. K. Belew. “No perfect two-state cellular automata for density classification exists”. In: *Physical Review Letters* 74.25 (1995), pp. 5148–5150.
- [LBC14] X. Lin, P. A. Beling, and R. Cogill. *Multi-agent inverse reinforcement learning for zero-sum games*. 2014. arXiv: 1403.6508.
- [LDK95] M. L. Littman, T. L. Dean, and L. P. Kaelbling. “On the complexity of solving Markov decision problems”. In: *Conference on Uncertainty in Artificial Intelligence*. 1995, pp. 394–402.
- [Lio+17] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters. “Learning movement primitive libraries through probabilistic segmentation”. In: *International Journal of Robotics Research* 36.8 (2017), pp. 879–894.

- [LL07] J.-M. Lasry and P.-L. Lions. “Mean field games”. In: *Japanese Journal of Mathematics* 2.1 (2007), pp. 229–260.
- [LMG05] K. Lerman, A. Martinoli, and A. Galstyan. “A review of probabilistic macroscopic models for swarm robotic systems”. In: *International Workshop on Swarm Robotics*. 2005, pp. 143–152.
- [LOT03] V. Lesser, C. L. Ortiz Jr, and M. Tambe. *Distributed Sensor Networks: A Multiagent Perspective*. Springer, 2003.
- [LP03] M. G. Lagoudakis and R. Parr. “Least-squares policy iteration”. In: *Journal of Machine Learning Research* 4.Dec (2003), pp. 1107–1149.
- [LPE00] E. Levin, R. Pieraccini, and W. Eckert. “A stochastic model of human-machine interaction for learning dialog strategies”. In: *IEEE Transactions on Speech and Audio Processing* 8.1 (2000), pp. 11–23.
- [LPK11] S. Levine, Z. Popovic, and V. Koltun. “Nonlinear inverse reinforcement learning with Gaussian processes”. In: *Advances in Neural Information Processing Systems*. 2011, pp. 19–27.
- [Mac+15] S. V. Macua, J. Chen, S. Zazo, and A. H. Sayed. “Distributed policy evaluation under multiple behavior strategies”. In: *IEEE Transactions on Automatic Control* 60.5 (2015), pp. 1260–1274.
- [Mac90] B. J. MacLennan. *Continuous spatial automata*. Tech. rep. University of Tennessee, 1990.
- [Mae+14] G. Maeda, M. Ewerton, R. Lioutikov, H. B. Amor, J. Peters, and G. Neumann. “Learning interaction for collaborative tasks with probabilistic movement primitives”. In: *IEEE/RAS International Conference on Humanoid Robots*. 2014, pp. 527–534.
- [MCH13] B. Michini, M. Cutler, and J. P. How. “Scalable reward learning from demonstration”. In: *IEEE International Conference on Robotics and Automation*. 2013, pp. 303–308.
- [Mel08] F. S. Melo. “Exploiting locality of interactions using a policy-gradient approach in multiagent learning”. In: *European Conference on Artificial Intelligence*. 2008, pp. 157–161.
- [Meu+99] N. Meuleau, L. Peshkin, K.-E. Kim, and L. P. Kaelbling. “Learning finite-state controllers for partially observable environments”. In: *Conference on Uncertainty in Artificial Intelligence*. 1999, pp. 427–436.

- [MH12] B. Michini and J. P. How. “Bayesian nonparametric inverse reinforcement learning”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2012, pp. 148–163.
- [Mic+15] B. Michini, T. J. Walsh, A. A. Agha-Mohammadi, and J. P. How. “Bayesian nonparametric reward learning from demonstration”. In: *IEEE Transactions on Robotics* 31.2 (2015), pp. 369–386.
- [MIM99] A. Martinoli, A. J. Ijspeert, and F. Mondada. “Understanding collective aggregation mechanisms: from probabilistic modelling to experiments with real robots”. In: *Robotics and Autonomous Systems* 29.1 (1999), pp. 51–63.
- [Miy+96] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato. “A Kendama learning robot based on bi-directional theory”. In: *Neural Networks* 9.8 (1996), pp. 1281–1302.
- [Mni+15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), pp. 529–533.
- [MT13] B. T. Morris and M. Trivedi. “Understanding vehicular traffic behavior from video: a survey of unsupervised approaches”. In: *Journal of Electronic Imaging* 22.4 (2013), p. 041113.
- [Mun06] R. Munos. “Policy gradient in continuous time”. In: *Journal of Machine Learning Research* 7.May (2006), pp. 771–791.
- [Nat+10] S. Natarajan, G. Kunapuli, K. Judah, P. Tadepalli, K. Kersting, and J. Shavlik. “Multi-agent inverse reinforcement learning”. In: *International Conference on Machine Learning and Applications*. 2010, pp. 395–400.
- [Nea00] R. M. Neal. “Markov chain sampling methods for Dirichlet process mixture models”. In: *Journal of Computational and Graphical Statistics* 9.2 (2000), pp. 249–265.
- [NHR99] A. Y. Ng, D. Harada, and S. J. Russell. “Policy invariance under reward transformations: theory and application to reward shaping”. In: *International Conference on Machine Learning*. 1999, pp. 278–287.
- [Nie+12] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto. “Learning and generalization of complex tasks from unstructured demonstrations”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5239–5246.

- [NLJ15] Q. P. Nguyen, B. K. H. Low, and P. Jaillet. “Inverse reinforcement learning with locally consistent reward functions”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 1747–1755.
- [NR00] A. Y. Ng and S. J. Russell. “Algorithms for inverse reinforcement learning”. In: *International Conference on Machine Learning*. 2000, pp. 663–670.
- [NS07] G. Neu and C. Szepesvári. “Apprenticeship learning using inverse reinforcement learning and gradient methods”. In: *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2007, pp. 295–302.
- [Oli12] F. A. Oliehoek. “Decentralized POMDPs”. In: *Reinforcement Learning 12* (2012), pp. 471–503.
- [OMT08] O. E. Omel’chenko, Y. L. Maistrenko, and P. A. Tass. “Chimera states: the natural link between coherence and incoherence”. In: *Physical Review Letters* 100.4 (2008), p. 044105.
- [Osa+18] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. “An algorithmic perspective on imitation learning”. In: *Foundations and Trends in Robotics* 7.1-2 (2018), pp. 1–179.
- [OSK08] J. Ohkubo, N. Shnerb, and D. A. Kessler. “Transition phenomena induced by internal noise and quasi-absorbing state”. In: *Journal of the Physical Society of Japan* 77.4 (2008), p. 044002.
- [PG17] A. Panella and P. Gmytrasiewicz. “Interactive POMDPs with finite-state models of other agents”. In: *Autonomous Agents and Multi-Agent Systems* 31.4 (2017), pp. 861–904.
- [PGP13] B. Piot, M. Geist, and O. Pietquin. “Learning from demonstrations: is it worth estimating a reward function?” In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2013, pp. 17–32.
- [PGP17] B. Piot, M. Geist, and O. Pietquin. “Bridging the gap between imitation learning and inverse reinforcement learning”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.8 (2017), pp. 1814–1826.
- [Pie13] O. Pietquin. “Inverse reinforcement learning for interactive systems”. In: *Workshop on Machine Learning for Interactive Systems: Bridging the Gap Between Perception, Action and Communication*. 2013, pp. 71–75.
- [Pom91] D. A. Pomerleau. “Efficient training of artificial neural networks for autonomous navigation”. In: *Neural Computation* 3.1 (1991), pp. 88–97.

- [Pot52] R. B. Potts. “Some generalized order-disorder transformations”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 48.1 (1952), pp. 106–109.
- [Pre+07] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.
- [Put94] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.
- [PVS03] J. Peters, S. Vijayakumar, and S. Schaal. “Reinforcement learning for humanoid robotics”. In: *IEEE/RAS International Conference on Humanoid Robots*. 2003, pp. 1–20.
- [PYG09] E. Papadimitriou, G. Yannis, and J. Golias. “A critical assessment of pedestrian behaviour models”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 12.3 (2009), pp. 242–255.
- [RA07] D. Ramachandran and E. Amir. “Bayesian inverse reinforcement learning”. In: *International Joint Conference on Artificial Intelligence* (2007), pp. 2586–2591.
- [Ras99] C. E. Rasmussen. “The infinite Gaussian mixture model”. In: *Advances in Neural Information Processing Systems*. 1999, pp. 554–560.
- [RBA05] W. Ren, R. W. Beard, and E. M. Atkins. “A survey of consensus problems in multi-agent coordination”. In: *American Control Conference*. 2005, pp. 1859–1864.
- [RD11] C. A. Rothkopf and C. Dimitrakakis. “Preference elicitation and inverse reinforcement learning”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2011, pp. 34–48.
- [Red+12] T. S. Reddy, V. Gopikrishna, G. Zaruba, and M. Huber. “Inverse reinforcement learning for decentralized non-cooperative multiagent systems”. In: *IEEE International Conference on Systems, Man, and Cybernetics*. 2012, pp. 1930–1935.
- [Ris96] H. Risken. “Fokker-planck equation”. In: *The Fokker-Planck Equation*. Springer, 1996, pp. 63–95.
- [Rot08] C. A. Rothkopf. “Modular models of task based visually guided behavior”. PhD thesis. University of Rochester, 2008.

- [RRK15] P. Ranchod, B. Rosman, and G. Konidaris. “Nonparametric Bayesian reward segmentation for skill discovery using inverse reinforcement learning”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2015, pp. 471–477.
- [RS97] G. O. Roberts and S. K. Sahu. “Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler”. In: *Journal of the Royal Statistical Society: Series B* 59.2 (1997), pp. 291–317.
- [RTG98] Y. Rubner, C. Tomasi, and L. J. Guibas. “A metric for distributions with applications to image databases”. In: *IEEE International Conference on Computer Vision*. 1998, pp. 59–66.
- [Rüc+13] E. Rückert, G. Neumann, M. Toussaint, and W. Maass. “Learned graphical models for probabilistic planning provide a new class of movement primitives”. In: *Frontiers in Computational Neuroscience* 6 (2013), p. 97.
- [RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press Cambridge, 2006.
- [Sam+92] C. Sammut, S. Hurst, D. Kedzier, and D. Michie. “Learning to fly”. In: *Machine Learning Proceedings*. 1992, pp. 385–393.
- [Sär13] S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.
- [SB98] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [Sch+05] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. “Learning movement primitives”. In: *International Symposium on Robotics Research*. 2005, pp. 561–572.
- [Sch03] F. Schweitzer. *Brownian agents and active particles: collective dynamics in the natural and social sciences*. Springer, 2003.
- [Sch99] S. Schaal. “Is imitation learning the route to humanoid robots?” In: *Trends in Cognitive Sciences* 3.6 (1999), pp. 233–242.
- [Seh+10] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber. “Parameter-exploring policy gradients”. In: *Neural Networks* 23.4 (2010), pp. 551–559.
- [Set12] B. Settles. “Active learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1 (2012), pp. 1–114.

- [Sil+16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (2016), pp. 484–489.
- [Sip99] M. Sipper. “The emergence of cellular computing”. In: *Computer* 32.7 (1999), pp. 18–26.
- [SK09] X. Su and T. M. Khoshgoftaar. “A survey of collaborative filtering techniques”. In: *Advances in Artificial Intelligence* (2009).
- [Šoš+17a] A. Šošić, W. R. KhudaBukhsh, A. M. Zoubir, and H. Koepl. “Inverse reinforcement learning in swarm systems”. In: *AAMAS Workshop on Transfer in Reinforcement Learning*. 2017.
- [Šoš+17b] A. Šošić, W. R. KhudaBukhsh, A. M. Zoubir, and H. Koepl. “Inverse reinforcement learning in swarm systems”. In: *International Conference on Autonomous Agents and Multiagent Systems*. **Best paper award finalist**. 2017, pp. 1413–1421.
- [Šoš+18] A. Šošić, E. Rückert, J. Peters, A. M. Zoubir, and H. Koepl. “Inverse reinforcement learning via spatio-temporal nonparametric subgoal modeling”. In: *Journal of Machine Learning Research* (2018). **Accepted**. arXiv: 1803.00444.
- [SP02] M. Stolle and D. Precup. “Learning options in reinforcement learning”. In: *International Symposium on Abstraction, Reformulation, and Approximation*. 2002, pp. 212–223.
- [SPS99] R. S. Sutton, D. Precup, and S. Singh. “Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning”. In: *Artificial Intelligence* 112.1 (1999), pp. 181–211.
- [SS08] U. Syed and R. E. Schapire. “A game-theoretic approach to apprenticeship learning”. In: *Advances in Neural Information Processing Systems*. 2008, pp. 1449–1456.
- [SS10] U. Syed and R. E. Schapire. “A reduction from apprenticeship learning to classification”. In: *Advances in Neural Information Processing Systems*. 2010, pp. 2253–2261.

- [SS14] A. Surana and K. Srivastava. “Bayesian nonparametric inverse reinforcement learning for switched Markov decision processes”. In: *IEEE International Conference on Machine Learning and Applications*. 2014, pp. 47–54.
- [ST11] M. M. So and L. C. Thomas. “Modelling the profitability of credit cards by Markov decision processes”. In: *European Journal of Operational Research* 212.1 (2011), pp. 123–130.
- [SWB05] Ö. Şimşek, A. P. Wolfe, and A. G. Barto. “Identifying useful subgoals in reinforcement learning by local graph partitioning”. In: *International Conference on Machine Learning*. 2005, pp. 816–823.
- [ŠZK16] A. Šošić, A. M. Zoubir, and H. Koepl. “Policy recognition via expectation maximization”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2016, pp. 4801–4805.
- [ŠZK18a] A. Šošić, A. M. Zoubir, and H. Koepl. “A Bayesian approach to policy recognition and state representation learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.6 (2018), pp. 1295–1308.
- [ŠZK18b] A. Šošić, A. M. Zoubir, and H. Koepl. “Inverse reinforcement learning via nonparametric subgoal modeling”. In: *AAAI Spring Symposium on Data-Efficient Reinforcement Learning*. 2018.
- [ŠZK18c] A. Šošić, A. M. Zoubir, and H. Koepl. “Reinforcement learning in a continuum of agents”. In: *Swarm Intelligence* 12.1 (2018), pp. 23–51.
- [Tam+15] M. Tamassia, F. Zambetta, W. Raffe, and X. Li. “Learning options for an MDP from demonstrations”. In: *Australasian Conference on Artificial Life and Computational Intelligence*. 2015, pp. 226–242.
- [TB08] A. Tewari and P. L. Bartlett. “Optimistic linear programming gives logarithmic regret for irreducible MDPs”. In: *Advances in Neural Information Processing Systems*. 2008, pp. 1505–1512.
- [Tes95] G. Tesauro. “Temporal difference learning and TD-Gammon”. In: *Communications of the ACM* 38.3 (1995), pp. 58–68.
- [THS01] L. C. Thomas, J. Ho, and W. T. Scherer. “Time will tell: behavioural scoring and the dynamics of consumer credit assessment”. In: *IMA Journal of Management Mathematics* 12.1 (2001), pp. 89–103.
- [TK14] H. M. Taylor and S. Karlin. *An Introduction to Stochastic Modeling*. Academic press, 2014.

- [Vic+95] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet. “Novel type of phase transition in a system of self-driven particles”. In: *Physical Review Letters* 75.6 (1995), pp. 1226–1229.
- [VZ12] T. Vicsek and A. Zafeiris. “Collective motion”. In: *Physics Reports* 517.3 (2012), pp. 71–140.
- [WD92] C. J. C. H. Watkins and P. Dayan. “Q-learning”. In: *Machine learning* 8.3-4 (1992), pp. 279–292.
- [WG02] G. M. Whitesides and B. Grzybowski. “Self-assembly at all scales”. In: *Science* 295.5564 (2002), pp. 2418–2421.
- [WO12] M. Wiering and M. van Otterlo. *Reinforcement Learning: State-of-the-Art*. Springer, 2012.
- [Yan+07] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo. “Evaluating bag-of-visual-words representations in scene classification”. In: *International Workshop on Multimedia Information Retrieval*. 2007, pp. 197–206.
- [Yan+12] S. Yang, M. Paddrik, R. Hayes, A. Todd, A. Kirilenko, P. Beling, and W. Scherer. “Behavior based learning in identifying high frequency trading strategies”. In: *IEEE Conference on Computational Intelligence for Financial Engineering & Economics*. 2012.
- [ZAS12] A. M. Zungeru, L.-M. Ang, and K. P. Seng. “Classical and swarm intelligence based routing protocols for wireless sensor networks: a survey and comparison”. In: *Journal of Network and Computer Applications* 35.5 (2012), pp. 1508–1536.
- [Zie+08] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. “Maximum entropy inverse reinforcement learning”. In: *AAAI Conference on Artificial Intelligence*. 2008, pp. 1433–1438.
- [ZJ12] S. Zhifei and E. M. Joo. “A review of inverse reinforcement learning theory and recent advances”. In: *IEEE Congress on Evolutionary Computation*. 2012, pp. 1–8.

Erklärungen laut Promotionsordnung

§8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 18. Juni 2018

